Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Detekce vzorů v obrazech a užití pro analýzu kulturního dědictví

Diplomová práce

Bc. Lukáš Fessl

Vedoucí práce: Ing. Jiří Jelínek, CSc.

České Budějovice 2016

Bibliografické údaje

Lukáš Fessl, 2016: Detekce vzorů v obrazech a užití pro analýzu kulturního dědictví [Patterns detection in paintings for analysis of cultural heritage. Mgr. Thesis, in Czech.] - 107p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic

Anotace:
Tato diplomová práce se zabývá vytvořením aplikace pro detekci vzorů v uměleckých dílech. Práce obsahuje teoretickou část zaměřenou na dvojici algoritmů, použitých pro vyhledávání, kterými jsou obecná Houghovo transformace a SURF algoritmus. Další část obsahuje návrh řešení a implementaci. Výsledkem práce je několik aplikací. Aplikace pro správu uměleckých děl, serverová aplikace sloužící pro vyhledávání a dvojice klientských aplikací. Jedna pro klasické počítače a druhá pro mobilní telefony s operačním systémem Android. V závěru práce je popsáno několik testů, které byly provedeny pro ověření funkčnosti výsledných aplikací.

Annotation:
This master's thesis deals with developement of application for patterns detection in paintings. This thesis has theoretical part focused to two algorithms for patterns detection. First algorithm is general Hough transform and second is SURF algorithm. The next part deals with proposal and implementation. The results are administration application, server application and two client applications, for classic computer and mobile phone with operating system Android. The end of thesis deals with tests which serves to verification functionality of applications.

Prohlašuji, že svoji **diplomovou** práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své **diplomové** práce, a to **v nezkrácené podobě** elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích, dne 20. 4. 2016 Podpis……

Poděkování

Děkuji panu Ing. Jiřímu Jelínkovi, CSc., za odborné vedení mé diplomové práce a pomoc při jejím zpracování.

Obsah

1 Úvod

Rozpoznáváním obrazu se zabývá vědní disciplína nazývající se počítačové vidění. Lze ji využít pro detekci, popisování objektů, které nás zajímají ve 2D a 3D obrazech, pro detekci různých událostí, modelování objektů nebo okolního světa a v neposlední řadě pro interakci mezi člověkem a strojem [2]. V posledních letech se tento obor velmi rozvíjí, především díky lehké dostupnosti a stále výkonnějším počítačům.

Podle [4] se lze v současné době setkat s počítačovým viděním v medicíně, kdy se na základě obrazových dat u mikroskopu, RTP, ultrazvukového vyšetření či tomografie obecně stanovuje diagnóza pacienta. Počítačové vidění tak vede ke zkvalitnění diagnostického procesu a patří dnes již neodmyslitelně k základním metodám používaných při vyšetření pacienta.

Dále, jak uvádí [4], je možné také se s ním setkat v průmyslu při získávání informací za účelem podpory výrobního či logistického procesu. Užívá se především v oblasti kontroly kvality, kde sleduje a kontroluje jednotlivé produkty, aby bylo možné předcházet vznikům defektů. V rámci řízení výroby slouží k monitoringu výrobků ve výrobních halách či řízení logistických toků.

Také, podle [4], se v poslední době velice rozmáhá počítačové vidění využívané v autonomních vozidlech, a to od bezpilotních letounů až po automobily. Úroveň autonomnosti se zde pohybuje od částečně autonomních do plně autonomních vozidel, ve kterých systémy počítačového vidění pomáhají řidiči nebo pilotovi lépe zvládat různé situace.

Tato práce má za úkol vytvořit aplikaci schopnou vyhledávat vzory v uměleckých dílech pro potřeby Filozofické fakulty v Českých Budějovicích. Pro vyhledávání jsou použity dvě metody. První metoda je obecná Houghova transformace, která je založena na matematickém popisu obrazu a vzoru. Druhá metoda je založena na detekci klíčových bodů. Variací této metody je několik, v závislosti na způsobu získání klíčových bodů. Ve výsledné aplikaci je využito metody SURF (Speeded-Up Robust Features). V závěru jsou tyto metody porovnány jak do rychlosti, tak i do přesnosti a výsledná aplikace nabízí možnosti použití obou vyhledávacích metod.

2 Cíle práce

1. Shromáždit a popsat potřeby procesu zkoumání kulturního dědictví z hlediska detekce vzorů v 2D obrazech.
2. Provést rešerši zaměřenou na detekci a identifikaci vzorů v obraze – metody, příklady užití, nástroje.
3. Výběr vhodné metody pro detekci a identifikaci vzorů ve 2D. Návrh, implementace a pilotní ověření vzorové aplikace pro detekci vzorů ve výtvarných dílech (obrazech).
4. Návrh řešení klient (Android) – server pro praktické použití detekce.
5. Testování aplikace v ostrém provozu se skutečnými daty. Vyhodnocení výsledků práce a doporučení pro další řešitele.

3 Zpracování obrazu

Zpracování obrazu je disciplína, která se zabývá zpracováním digitálních obrazových dat a lze se s ní setkat v nejrůznějších oblastech. Podle [2] lze postup zpracování a rozpoznávání obrazu rozdělit do několika kroků, kterými jsou snímání, digitalizace, předzpracování, segmentace a rozpoznávání (klasifikace). Rozdělení kroků ovšem není zcela jednoznačné, a tak lze v literatuře nalézt různá dělení. Záleží také na samotné aplikaci, zda budou všechny kroky zpracování obrazu provedeny.

3.1 Snímání obrazu a digitalizace

Základem pro zpracování a rozpoznávání obrazu je vlastní získání obrazu z reálného světa a jeho převod do digitální formy, která je vhodná pro uložení a další zpracování.

Jak je uvedeno v [2], snímáním obrazu se rozumí převod optické veličiny na elektrický signál, který je spojitý v čase i úrovni. Na výsledný sejmutý obraz má vliv mnoho různých faktorů. Může to být například ozáření snímaného objektu a jeho vlastnosti. Vstupní informací při snímání nemusí být vždy jen jas z kamery či skeneru, ale mohou to být i jiné veličiny jako jsou intenzita rentgenového záření, ultrazvuk či tepelné záření.

Dalším krokem je převedení analogového signálu na digitální a to tak, aby byl vhodný pro zpracování počítačem. Výsledný digitální obraz [3] může být definován jako dvoudimenzionální funkce $f(x, y)$, kde x a y jsou prostorové koordináty, a amplituda f na jakékoliv dvojici souřadnic (x, y) je nazývána intenzitou nebo úrovní šedi obrazu v daném bodě. Digitální obraz je složen z konečného množství elementů a každý z nich má přesnou polohu a hodnotu. Tyto elementy se nazývají obrazové prvky, obrazové body nebo pixely.

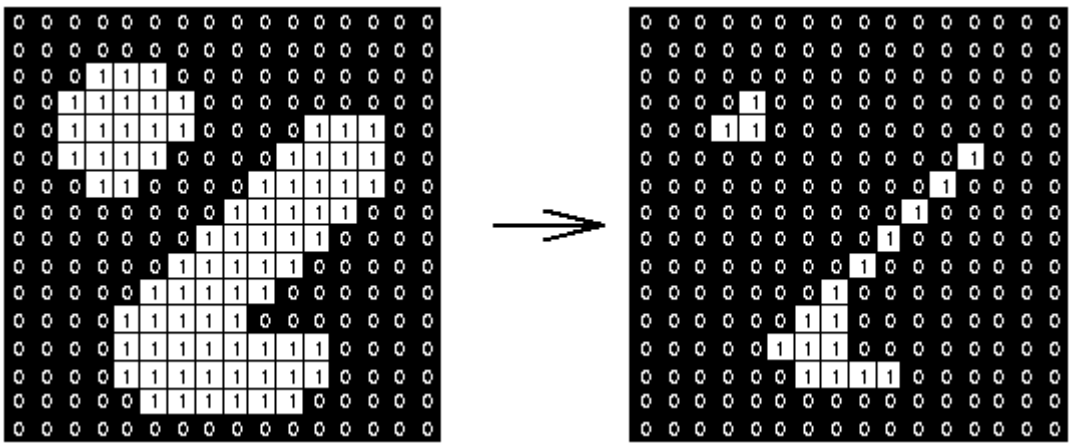
3.2 Předzpracování obrazu

Jak je uvedeno v [3], metody předzpracování obrazu slouží ke zlepšení obrazu z hlediska dalšího zpracování. Cílem předzpracování je potlačit šum vzniklý při digitalizaci a přenosu obrazu, odstranit zkreslení dané vlastnostmi snímacího zařízení, případně potlačit nebo zvýraznit jiné rysy důležité z hlediska dalšího zpracování. Mezi základní metody předzpracování obrazu je možné zahrnout převedení na stupně šedi [2], jasovou korekci pro opravu jasu v obrazu [4], ekvalizaci histogramu pro zlepšení kontrastu [5], ostření obrazu, filtraci pro odstranění statického šumu [6] či matematickou morfologii [7].

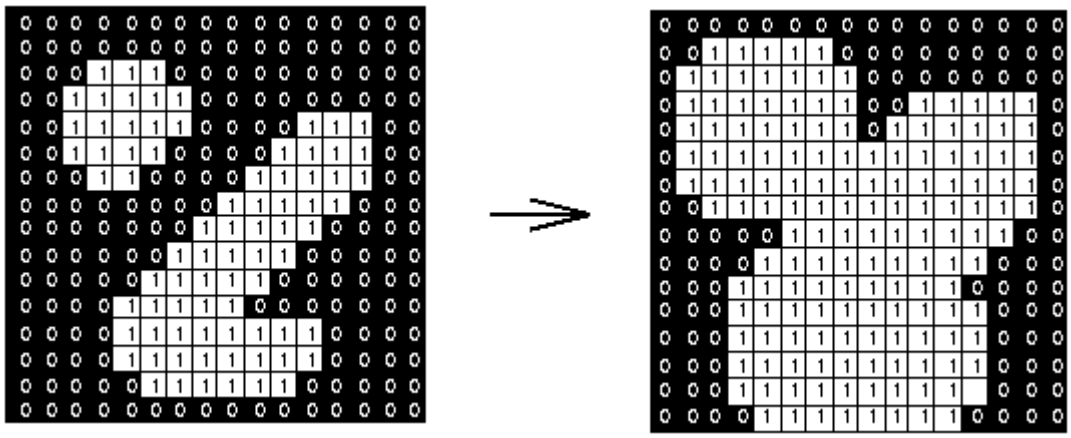
3.2.1 Eroze a dilatace

Podle [8], eroze a dilatace patří mezi základní techniky matematické morfologie. Eroze je založena na odstraňování pixelů z krajů objektu, zatímco dilatace body přidává. Počet odebraných nebo přidaných pixelů závisí na velikosti a tvaru masky, která je použita pro zpracování obrazu. Eroze se používá ke zjednodušení objektů a odstranění detailů z obrazu.

Právě proto je eroze využita ve výsledné aplikaci. Výsledný obraz je také menší než původní. Dilataci lze použít k zaplnění menších děr v objektech a výsledkem je větší objekt, než objekt původní. Příklad použití eroze a dilatace je znázorněn na obrázcích 3.1 a 3.2.



Obrázek 3.1 Efekt eroze s velikostí masky 3x3 na binárním obrazu [9]



Obrázek 3.2 Efekt dilatace s velikostí masky 3x3 na binárním obrazu [9]

3.3 Segmentace

Segmentace obrazu má za cíl rozdělit obraz do oblastní, které jsou stejné vzhledem k nějakému kritériu. Typickou úlohou segmentace je odlišit objekty od sebe, nebo od pozadí. Samotný proces segmentace není jednoduchý, protože již pořízená obrazová data jsou obvykle deformována šumem či nehomogením osvětlením. Objekty mohou být složité, nebo se mohou překrývat atd. V takovém případě je rozlišení jednotlivých objektů obtížné i pro lidské oko, natož pro počítačový algoritmus. Existuje celá řada metod, které lze při segmentaci využít. Mezi některé metody pro segmentaci obrazu patří prahování [10], hranové detektory [12], které mohou být založeny na první a druhé derivaci [14], techniky založené na regionech, Kohenenovy mapy či hybridní metody kombinující jednu nebo více základních segmentačních metod. Více o segmentačních metodách lze nalézt [14].

3.3.1 Prahování

Nejjednodušším přístupem k segmentaci obrazu je pomocí metody nazývané prahování [10], které je založeno na intenzitě obrazu. Tato technika rozděluje obraz do dvou skupin a pracuje s předpokladem, že pixely patřící do určitého rozsahu intenzit reprezentují jednu skupinu a zbytek pixelů v obraze reprezentuje druhou skupinu. Prahování může být globální nebo lokální.

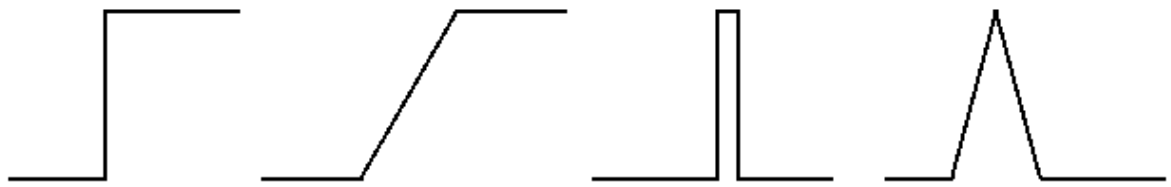
Globální prahování odlišuje pixely objektů a pozadí porovnáním se zvolenou prahovou hodnotou a použije se binární hodnoty pro oddělení objektů v obraze. Pixely, které projdou prahováním, jsou považovány za pixely objektu a je jim přiřazena binární hodnota 1. Ostatním pixelům je přiřazena binární hodnota 0 a jsou zpracované jako pixely pozadí. Tato technika segmentace není náročná, výpočetně je velmi rychlá a může být použita v „real-time“ aplikacích.

Lokální prahování je také známé jako adaptivní. V této technice se prahová hodnota mění přes celý obraz v závislosti na lokálních charakteristikách jednotlivých regionů v obraze. Algoritmus pro adaptivní prahování může být rozdělen do několika kroků. Rozdělení obrazu do regionů. Vybrání prahů pro jednotlivé regiony a nakonec provedení prahování pro každý region s jejím konkrétním prahem. Prahy pro jednotlivé regiony jsou vybrány v závislosti na vlastnostech pixelů daných regionů.

3.3.2 Detekce hran

Detekce hran je založena na hledání různých intenzit mezi pixely. Hrany jsou většinou nalezeny aplikováním konvolučního filtru [13] přes celý obraz. Hranou se rozumí takový bod v obrazu, u kterého se hodnota jasu prudce mění. Takováto hrana se nazývá skoková. V reálných obrazech je ale často změna jasu postupná, nikoli skoková. Takováto hrana se nazývá šikmá. Zkombinováním těchto dvou hran mohou vzniknout další dvě. Ty se nazývají čára a střecha. Jednotlivé typy hran jsou znázorněny na obrázku 3.3.

Hranové detektory mohou být založeny na první [14], nebo na druhé derivaci [14].



Obrázek 3.3 Typy hran v obraze. Zleva: skoková, šikmá, čára a střecha

3.3.3 Cannyho hranový detektor

Jak je uveden [8], Cannyho hranový detektor je algoritmus, který se skládá z několika kroků tak, aby bylo dosaženo co nejlepších výsledků při detekci hran.

Prvním krokem v tomto algoritmu je detekce a odstranění šumu v obraze. To se nejčastěji provádí pomocí Gaussova filtru, ale může být použit i jiný filtr. Důležité je správně nastavit vybraný filtr. Jeho špatné nastavení může způsobit, že budou detektorem detekovány i nevýznamné hrany, nebo také může dojít k zaniknutí slabých hran.

Po odstranění šumu následuje aplikování hranového detektoru. Nejvhodnější je použití tzv. Sobelova operátoru, ale i zde může být použit jiný hranový detektor. Použití Sobelova operátoru je nejvhodnější, protože není příliš citlivý na šum a vrací nejen velikost gradientu, ale i jeho směr.

Posledním krokem je prahování. Jeho použití má za cíl odstranit bezvýznamné hrany. To jsou hrany, které jsou krátké, nebo slabé. Toho ovšem nelze dosáhnout prahováním s jedním prahem. Proto jsou voleny prahy dva, minimální a maximální, mezi kterými může hodnota gradientu kolísat.

3.4 Metody rozpoznávání obrazu

Identifikace objektů je posledním krokem při rozpoznávání obrazu. K dispozici je celá řada přístupů a metod. Často používaná metoda je metoda založena na detekci klíčových bodů v obraze (Feature detection) [15]. Tyto body jsou pak dále popsány pomocí deskriptorů. Existuje několik variací této metody, a to podle způsobu hledání klíčových bodů a způsobu výpočtu deskriptorů, např. SIFT, SURF apod. Další způsob pro identifikaci objektů je Houghova transformace [16]. Ta je určena především pro hledání objektů, které je možné analyticky popsat, jako jsou přímky či kružnice. Pro identifikace složitějších objektů, které nelze analyticky popsat lze využít obecnou Houghovo transformaci, jejíž princip vychází ze starší metody nazývané Match template. Pro identifikaci objektů lze také využít metodu, založenou na výpočtu momentových příznaků objektu [17]. Další způsob pro identifikaci objektů je s využitím umělých neuronových sítí [17]. Tato metoda je založena na učení, při němž se neuronová síť postupně co nejlépe adaptuje na řešení daného problému. K dispozici je ale i celá řada dalších metod. V rámci aplikace je využito obecné Houghovy transformace a metody založené na detekci klíčových bodů realizované pomocí SURF algoritmu, a proto budou v následujících dvou kapitolách podrobněji popsány.

4 Houghova transformace

Houghova transformace slouží pro hledání objektů v obraze, které lze parametricky popsat. Pro hledání tvarů, které lze popsat snadno, jako jsou přímky, kružnice apod., slouží klasická Houghova transformace. Později vznikla obecná Houghova transformace, která se používá pro detekci vzorů v obraze, které nelze parametricky popsat.

Pro každý obraz je vytvořen tzv. Houghův prostor nebo častěji akumulátor, jehož rozměry jsou totožné s velikostí obrazu. Do něj se zanáší hodnoty pro každé umístění vzoru v obraze, jejichž hodnota je dána shodou hledaného vzoru v obraze. Hlavní princip tedy spočívá v postupném posouvání hledaného vzoru po obraze a u každého umístění dochází s porovnáním všech bodů vzoru se všemi body v obraze. Tato metoda je tak určitou obdobou brute-force algoritmu, z čehož vyplývá i nevýhoda, a to je velká výpočetní náročnost. Další nevýhodou může být to, že Houghova transformace není invariantní vůči velikosti a natočení a mohou tak vznikat až 4 rozměrné akumulátory, z čehož vyplývá i velká paměťová náročnost. Je tak vhodnější používat tuto metodu pro potřeby, kde se invariance nemusí řešit. Hlavní výhodou této metody je poměrně velká odolnost na různé druhy šumu či nepravidelnosti hledaného vzoru v obraze.

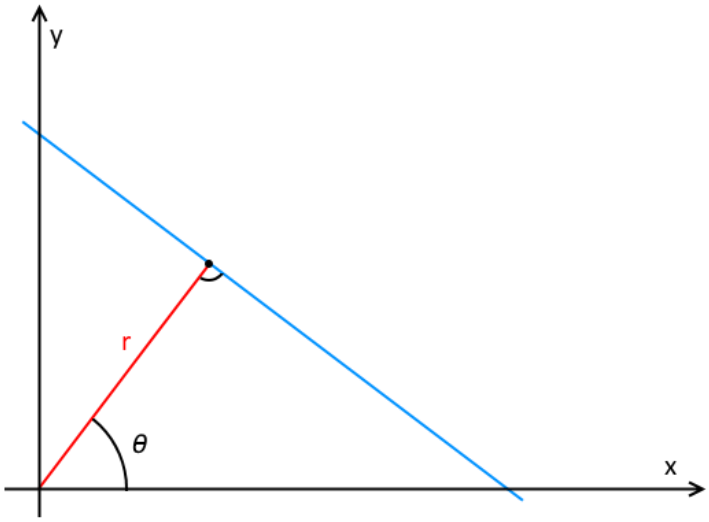
4.1 Detekce přímek

Nejjednodušší případ využití klasické Houghovy transformace je pro detekci přímek. V obrazovém prostoru může být přímka popsána vzorcem (1).

$$y = ax + b \tag{1}$$

Používání tohoto vzorce má však své nevýhody, a to pro přímky, které jsou ve vertikální nebo horizontální poloze. Proto se doporučuje použít vzorec (2) pro nalezení pozice přímky, kde r je vzdálenost od počátku souřadnic k hledané přímce, viz obrázek 4.1. Tato hledaná přímka je vždy kolmá k přímce r. Úhel θ (theta) je úhel mezi osou x a přímkou r. Volí se krok změny úhlu θ a ke každému úhlu se dopočítává velikost r.

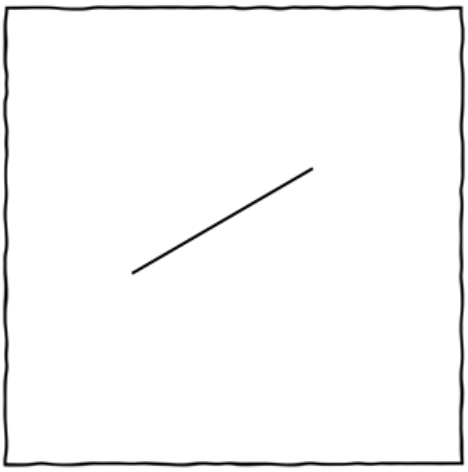
$$r = x * \cos\theta + y * \sin\theta \tag{2}$$



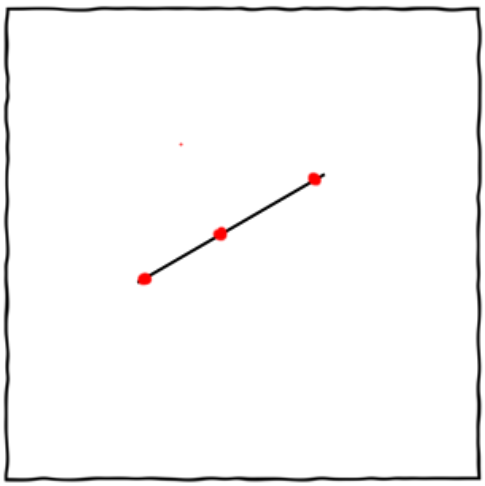
Obrázek 4.1 Grafické znázornění vzorce (2)

Je tedy možné logicky spojit každou přímku *r* a úhel *θ*. Z těchto dvojic *(r, θ)* se poté vytváří akumulátor. Princip detekce přímek je popsán na sérii následujících obrázků.

Nejprve musí obrázek projít detektorem hran. Pro zjednodušení jsou na obrázku zachyceny jen 3 body reprezentující přímku. Ve skutečnosti je těchto bodů mnohem více.

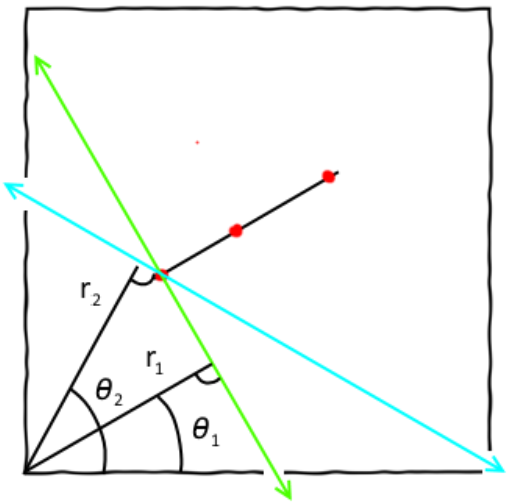


Obrázek 4.2 Vzorový obrázek

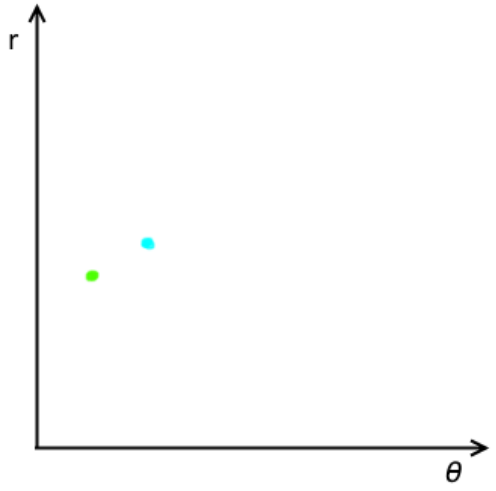


Obrázek 4.3 Detekované body

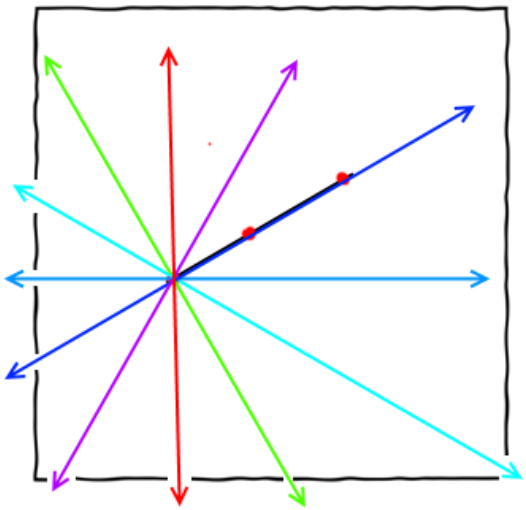
Je zvolen krok, pro změnu úhlu θ a následně jsou konstruovány imaginární přímky procházející prvním nalezeným bodem, viz obrázek 4.4. Těmto úhlům jsou dopočítávány vzdálenosti r podle vzorce (2). Výsledné páry (r, θ) jsou zaneseny do akumulátoru, viz obrázek 4.5.



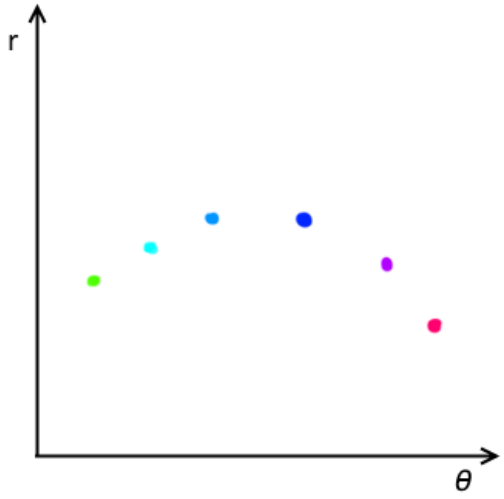
Obrázek 4.4 Dvě přímky procházející prvním detekovaným bodem



Obrázek 4.5 Vynesení dvou párů (r, θ) do grafu reprezentující akumulátor

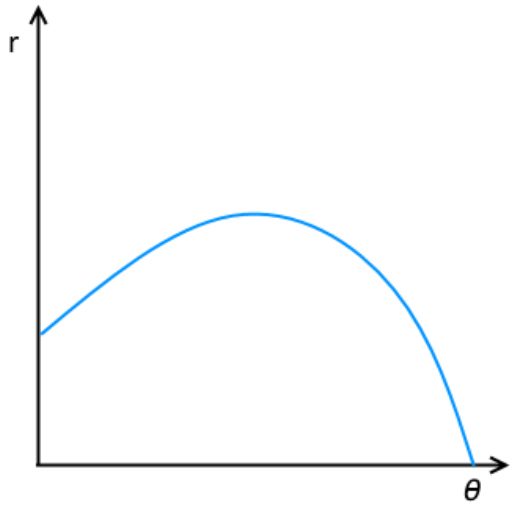


Obrázek 4.6 Všechny přímky procházející prvním bodem

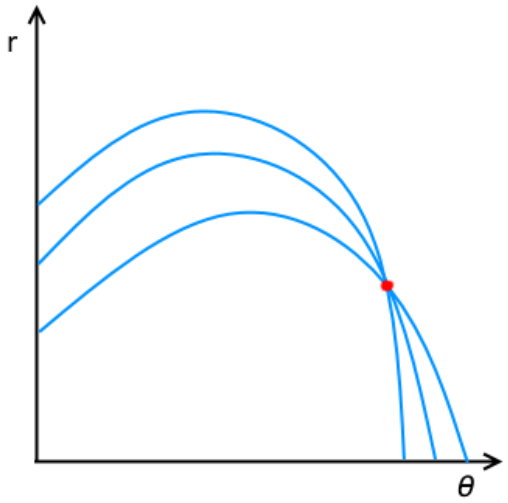


Obrázek 4.7 Vynesení všech párů (r, θ) do grafu reprezentující akumulátor

Všechny možné přímky, které prochází prvním bodem, je možné reprezentovat v akumulátoru jako křivku, viz obrázek 4.8. Dále se opakují předchozí kroky pro každý nalezený bod. Výsledný akumulátor je možné vidět na obrázku 4.9. Každá křivka zde reprezentuje právě jeden bod. Místo kde se protíná nejvíce křivek (vyznačeno červeně) identifikuje přímku na původním obrázku. Z akumulátoru tedy vyčteme vzdálenost od počátku souřadnic a úhel, potřebný k nalezení přímky. Více o detekci přímek lze nalézt zde [16], [18] a [19].



Obrázek 4.8 Všechny přímky procházející prvním bodem



Obrázek 4.9 Všechny přímky procházející všemi body

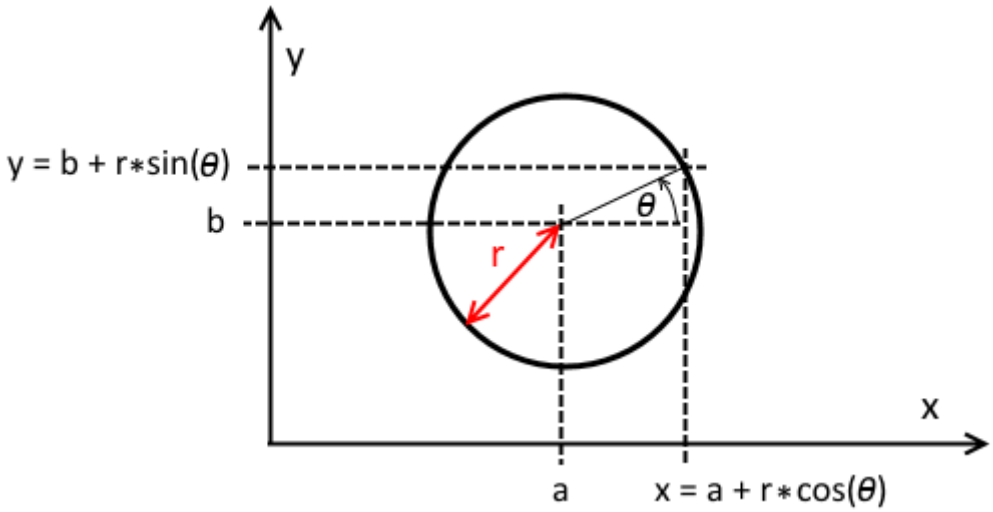
Obrázek 4.10 Příklad skutečného akumulátoru [19]

4.2 Detekce kružnic

Pro detekci kružnic se používá obdobná myšlenka jako pro detekci přímek. Zde nejlépe popisuje kružnici parametrické vyjádření pro parametr *x* vzorec (3) a pro parametr *y* vzorec (4). Parametry *x*, *y* jsou body na kružnici, parametry *a*, *b* označují střed kružnice, *r* je poloměr kružnice a θ je úhel od 0° do 360°. Jednotlivé vzorce jsou znázorněny na obrázku 4.11.

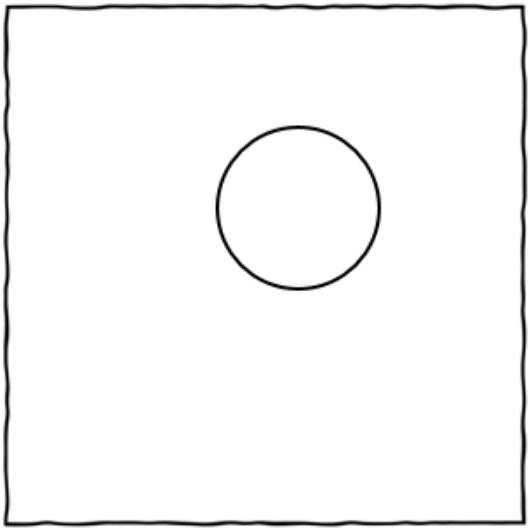
$$x = a + r * \cos\theta \tag{3}$$

$$y = b + r * \sin\theta \tag{4}$$

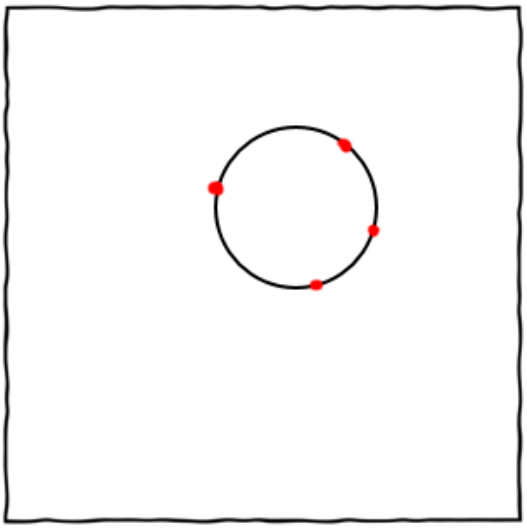


Obrázek 4.11 Parametrické vyjádření kružnice

Pro detekci kružnic je nejprve nutné obrázek opět předzpracovat a vyhledat v něm hrany, které reprezentují danou kružnici. V tomto případě hrany reprezentuje čtveřice červených bodů.

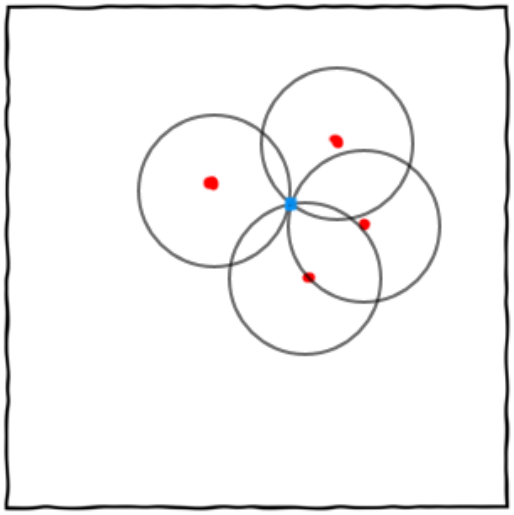


Obrázek 4.12 Vzorový obrázek

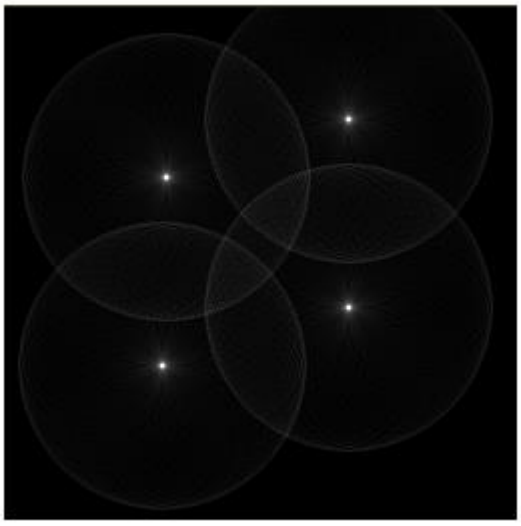


Obrázek 4.13 Detekované body

Na každý nalezený bod jsou aplikovány vzorce (3) a (4), kdy nalezený bod je středem imaginární kružnice. Tato imaginární kružnice se vynáší do akumulátoru, jako tomu bylo u přímky. Na osách akumulátoru není ovšem tentokrát úhel a poloměr, ale souřadnice *x*, *y*. V místě akumulátoru, kde dojde k nejvíce překrytí, bude střed kružnice.

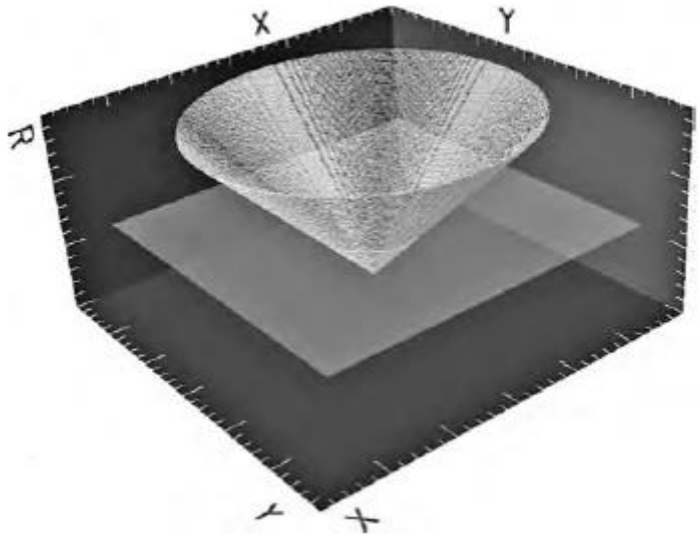


Obrázek 4.14 Výsledný akumulátor



Obrázek 4.15 Příklad skutečného akumulátoru

Toto platí v případě, že je znám poloměr kružnice. Pokud ovšem poloměr kružnice znám není, je třeba za poloměr dosadit vhodné rozmezí hodnot, akumulátor tak bude trojrozměrný, viz obrázek 4.16. Z toho vyplývá, že se bude výrazně zvedat výpočetní i paměťová náročnost. Je proto třeba zvolit výhodný rozsah poloměrů, protože příliš velký rozsah prodlouží výpočet, a příliš malý rozsah může zapříčinit, že kružnice nebude nalezena. Více o detekci kružnic pomocí klasické Houghovy transformace lze nalézt zde [16] a [18].



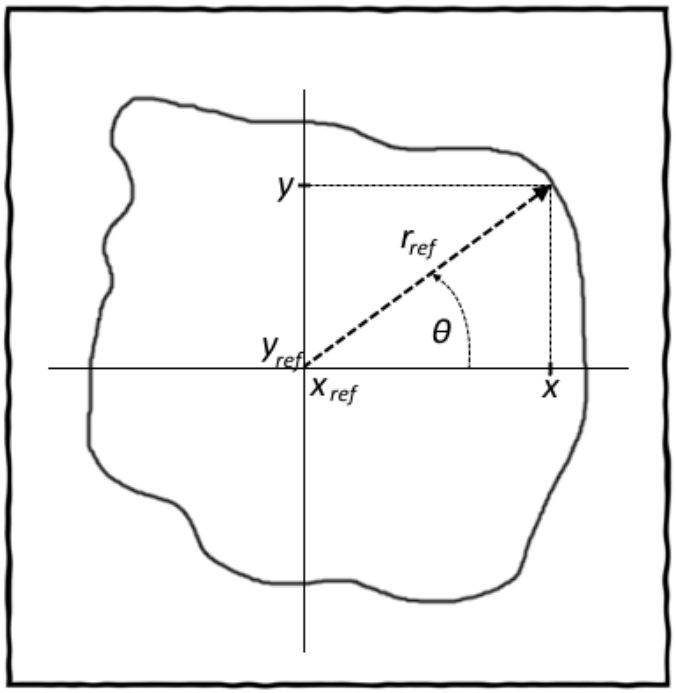
Obrázek 4.16 Trojrozměrný akumulátor pro jeden bod a pro různá *r* [16].

4.3 Obecná Houghova transformace

Při hledání tvaru, který je tak složitý, že ho není možné popsat parametricky, lze využít obecné Houghovy transformace. Místo rovnice popisující hledaný tvar je zde použita tzv. referenční tabulka, která tento nepravidelný vzor popisuje. Pro sestavení referenční tabulky je třeba vzor, který se zpracuje tak, aby na něm vznikly hrany. Poté je určen referenční bod. To je imaginární bod, od kterého se počítá vzdálenost *r* ke každému bodu reprezentující vzor, podle vzorce (5), a jeho úhel θ svírající s osou *x*, podle vzorce (6). Získané vzdálenosti a úhly jsou poté zaneseny do referenční tabulky, přičemž jednomu úhlu může připadnout i více vzdáleností v závislosti na složitosti objektu. Pozice referenčního bodu je volitelná, ale pro co nejlepší výsledek je vhodné tento bod umístit do středu objektu, který lze získat např. zprůměrováním pozic všech bodů reprezentující vzor.

$$r^2 = (x - x_{ref})^2 + (y - y_{ref})^2 \tag{5}$$

$$\theta = \arctan \frac{(y - y_{ref})}{(x - x_{ref})} \tag{6}$$



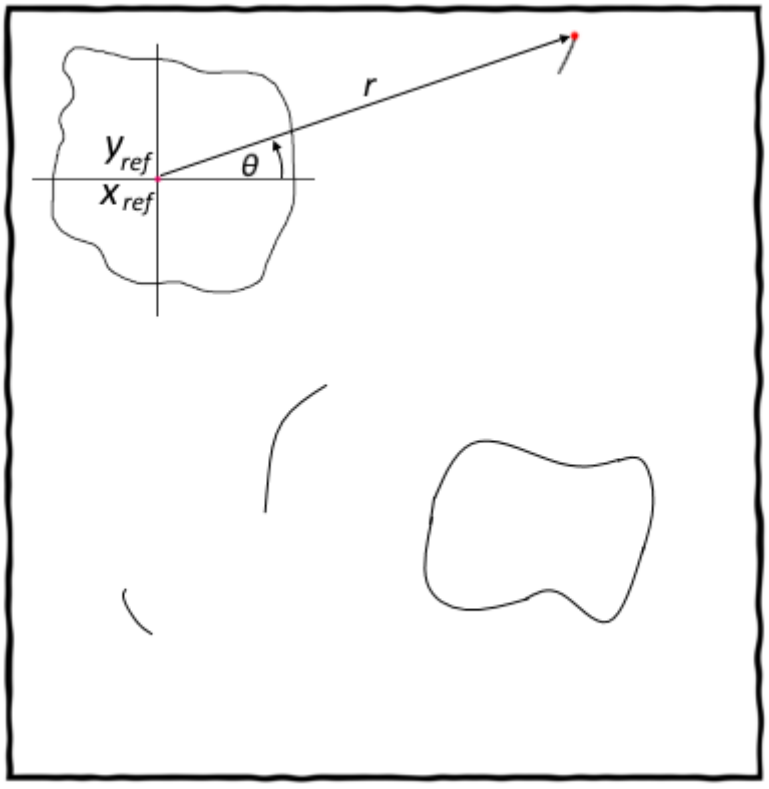
Obrázek 4.17 Obrázek reprezentující vzor

θ	r_{ref}
0	38
1	37
2	36
3	34
4	34
5	35
6	36
7	38
8	39
…	…

Tabulka 4.18 Příklad referenční tabulky

Před vyhledáváním je na obraz aplikován hranový detektor pro detekci hran. Při vyhledávání je vzor umístěn do levého horního rohu obrazu. Postupně se prochází nalezené body v obraze reprezentující hrany. Pro daný bod v obraze je spočítán úhel θ, který svírá s osou x u vzoru a vzdálenost r_{ref} od referenčního bodu vzoru, viz obrázek 4.19. Na základě vypočteného úhlu θ je vybrán z referenční tabulky záznam s příslušným úhlem a tak i příslušná vzdálenost r. Tyto hodnoty jsou poté použity ve vzorci (7) a výsledná hodnota je zanesena do akumulátoru. Tato hodnota se pohybuje od 0 do 1, kdy hodnota 1 vyjadřuje, že poloha bodu ve vzoru se nachází přesně na bodu v obraze, se kterým se počítá. S rostoucí vzdáleností mezi body klesá tato hodnota až k nule.

$$a = e^{-\frac{(r - r_{ref})^2}{2 * \sigma^2}} \tag{7}$$



Obrázek 4.19 Vyhledávání vzoru v obraze

Parametr σ (sigma) určuje toleranci nepřesnosti. To znamená např. pokud vzdálenost mezi r a r_{ref} je 5 a σ je 2, pak výsledek vzorce (7) je 0,04. Tato hodnota říká, že nastavená tolerance pro nepřesnost je velice malá a pro úspěšnou identifikaci vzoru bude muset hledaný tvar

v obraze být téměř stejný. Pokud, naopak σ má hodnotu 25, pak výsledek vzorce (7) je 0,99. Tato hodnota tak říká, že nastavení parametru σ je velice benevolentní na nepřesnost a velice pravděpodobně dojde k chybné identifikaci vzoru v obraze. Je tedy třeba zvolit správnou velikost pro parametr σ.

Tyto výpočty se provádí postupně pro každý bod v obraze a výsledná hodnota ze vzorce (7) se vždy přičte k hodnotě akumulátoru pro dané umístění vzoru v obraze. Po provedení výpočtů se všemi body v obraze je vzor posunut a dochází k opětovným výpočtům pro všechny body v obraze, ale pro nové umístění vzoru, který se takto postupně proloží celým obrazem.

U klasické Houghovy transformace pro hledání kružnic vznikl trojrozměrný akumulátor, a to v případě, že nebyla známa velikost hledaného kruhu. U obecné Houghovy transformace může být neznámá nejen velikost, ale i natočení hledaného vzoru. Pokud je třeba, aby obecná Houghova transformace byla invariantní vůči velikosti i natočení, vznikne tak 4 rozměrný akumulátor a počet potřebných výpočtů výrazně vzroste. Více informací o obecné Houghovo transformaci lze získat zde [20].

5 SURF

SURF (Speeded Up Robust Features) [21] je jeden ze způsobů, které se používají v metodě pro detekci klíčových bodů v obraze. Tato metoda vznikla v roce 2006 a vychází ze starší metody SIFT (The Scale Invariant Feature Transform) [22]. Oproti ní je SURF výrazně rychlejší a poskytuje přinejmenším stejně dobré výsledky. Prvním krokem této metody je nalezení bodů v obraze, které reprezentují zajímavá místa. Těmito místy jsou především hrany, rohy či větší místa se stejnou barevnou plochou. Pro zrychlení detekce zajímavých míst je využit tzv. Integrální obraz. Pro každý nalezený bod v obraze je vypočítán deskriptor, který popisuje daný bod. Významné body a deskriptory jsou vypočítány také pro vzor a na základě deskriptorů dochází k porovnávání. Metoda tedy nepopisuje obrázek jako celek, ale jako množinu v něm nalezených bodů. Hlavní výhody oproti obecné Houghovo transformaci, která byla zmíněna výše, je především vyšší rychlost a invariance vůči velikosti a natočení.

5.1 Integrální obraz

Integrální obraz [23] se využívá u SURF algoritmu pro detekci klíčových bodů, kdy je třeba spočítat plochu intenzity kolem nějakého bodu. Jedná se o velmi rychlý způsob získání potřebné hodnoty, protože stačí znát pouze čtyři hodnoty, a to z rohů dané plochy z integrálního obrazu. Nemusí se tak procházet celá plocha. Prvním krokem je výpočet integrálního obrazu z originálního obrazu. To se provádí tak, že každý pixel obsahuje hodnotu součtu intenzit bodů od počátku obrazu až do daného bodu, viz obrázek 5.1 a 5.2. Matematický zápis integrálního obrazu je popsán vzorcem (8), kde $I_{\sum}(x,y)$ je pozice v integrálním obrazu a $I\ (i,j)$ je pozice v originálním obrazu.

$$I_{\sum}(x,y) = \sum_{i=0}^{i \le x} \sum_{y=0}^{j \le y} I\ (i,j) \tag{8}$$

5	2	5	2
3	6	3	6
5	2	5	2
3	6	3	6

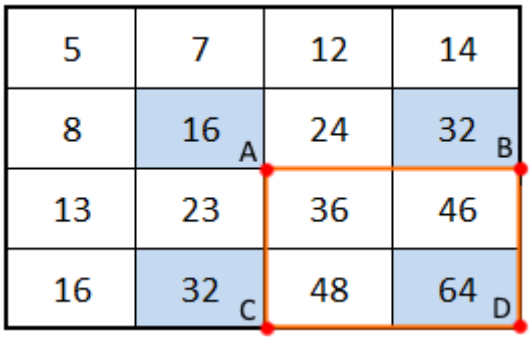
Obrázek 5.1 Originální obraz

5	7	12	14
8	16	24	32
13	23	36	46
16	32	48	64

Obrázek 5.2 Integrální obraz

Po výpočtu integrálního obrazu je možné získat součet hodnot vybrané plochy z obrazu pomocí vzorce (9), kde každá proměnná představuje právě jeden roh z vybrané plochy, viz obrázek 5.3.

$$\sum = I_{\sum}(A) + I_{\sum}(D) - I_{\sum}(C) - I_{\sum}(B) \tag{9}$$



Obrázek 5.3 Výběr plochy pro výpočet

Po dosazení do vzorce (9) příslušné hodnoty bodů z integrálního obrazu lze získat hodnotu 16, která odpovídá součtu hodnot té samé plochy v originálním obrazu.

5.2 Detekce významných bodů

Pro detekci významných bodů se používá determinant Hessovy matice. Jak uvádí [24], jedná se o čtvercovou matici druhých parciálních derivací skalární funkce. Determinant Hessovy matice pro bod o souřadnicích *x* a *y* se spočítá podle vzorce (10).

$$H(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \tag{10}$$

kde

$$L_{xx}(x, y, \sigma) = I(x, y) * \frac{\partial^2}{\partial x\mathrm{x}}\, g\,(\sigma) \tag{11}$$

$$L_{xy}(x, y, \sigma) = I(x, y) * \frac{\partial^2}{\partial \mathrm{xy}}\, g\,(\sigma) \tag{12}$$

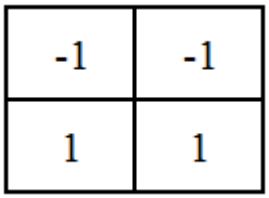
$L_{xx}(x, \sigma)$ u vzorce (11) je konvoluce obrazu v daném bodě s druhou derivací Gaussovy funkce $\frac{\partial^2}{\partial \mathrm{xy}}\, g\,(\sigma)$. $I(x, y)$ je hodnota z integrálního obrazu ze souřadnic *x* a *y*, kde je uchován součet hodnot z dané plochy, viz vzorec (8). Přičemž parametr σ umožňuje měnit zvětšení plochy, na kterou je determinant počítán. Podle [25], zvětšení plochy Gaussovského kernelu, vyjadřující jednu z derivací, je matematicky ekvivalentní výpočtu této derivace na úměrně zmenšeném obraze. Jelikož SURF používá k výpočtu konvolucí obrazu s Gaussovskými kernely integrální obraz, je výpočet stejně dlouhý pro libovolnou velikost Gaussovského kernelu, tudíž i úroveň zvětšení, na které je počítán. Díky této optimalizaci dosahuje algoritmus SURF značného zrychlení oproti jiným metodám, protože většina algoritmů v tomto místě provádí změnu velikosti obrazu převzorkováním.

5.3 Invariance vůči rotaci

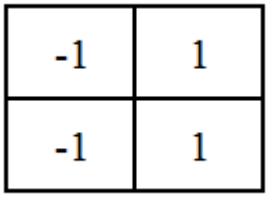
Aby byl SURF algoritmus invariantní vůči natočení, je třeba identifikovat orientaci samotných významných bodů. Jak je uvedeno v [18], pro každý významný bode je stanovena kruhová oblast o velikosti 6*s*, kde číslo znamená počet bodů v obraze a *s* udává měřítko zvětšení obrazu, ve kterém byl významný bod nalezen

Dále jsou v této kruhové oblasti aplikovány Haarovo vlnkové filtry, které jsou znázorněny na obrázkách 5.4 a 5.5. Velikost filtrů je 4*s*. Výsledkem je nalezení gradientu ve směru osy *x* a *y*. Tyto hodnoty jsou dále váženy pomocí Gaussova filtru se středem na pozici významného bodu. Velikost Gaussova filtru je stanovena na 2,5*s*. Vypočtené hodnoty jsou poté reprezentovány jako body v prostoru, viz červené body v obrázku 5.6.

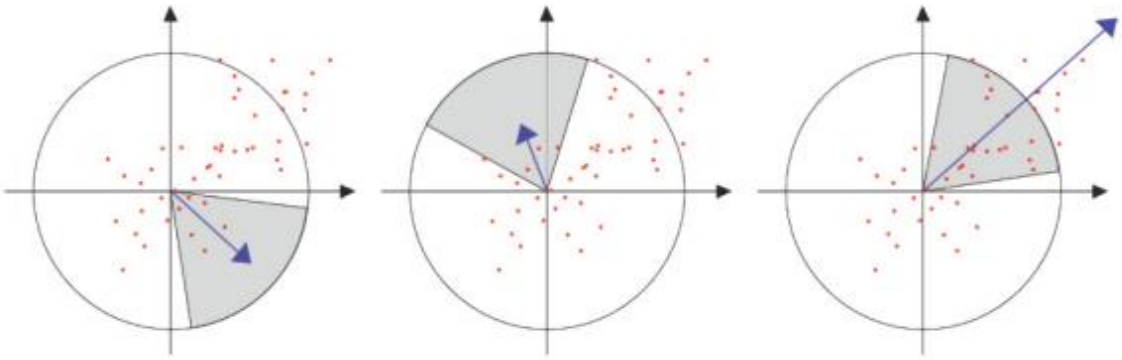
Pro stanovenou kruhovou oblast je definována kruhová výseč π/3, ve které se sečtou všechny body a vytvoří se z nich vektor, viz obrázek 5.6. Kruhová výseč postupně rotuje kolem významného bodu a pro každou pozici je vytvořen nový vektor. Jak je uvedeno v [22], krok by neměl být větší než 15°. Poté je vybrán největší vektor pro daný významný bod, který se nazývá dominantní. Deskriptor je poté počítán ve směru úhlu dominantního vektoru.



Obrázek 5.4 Filtr Haarových vlnek ve směru osy *x*



Obrázek 5.5 Filtr Haarových vlnek ve směru osy *y*



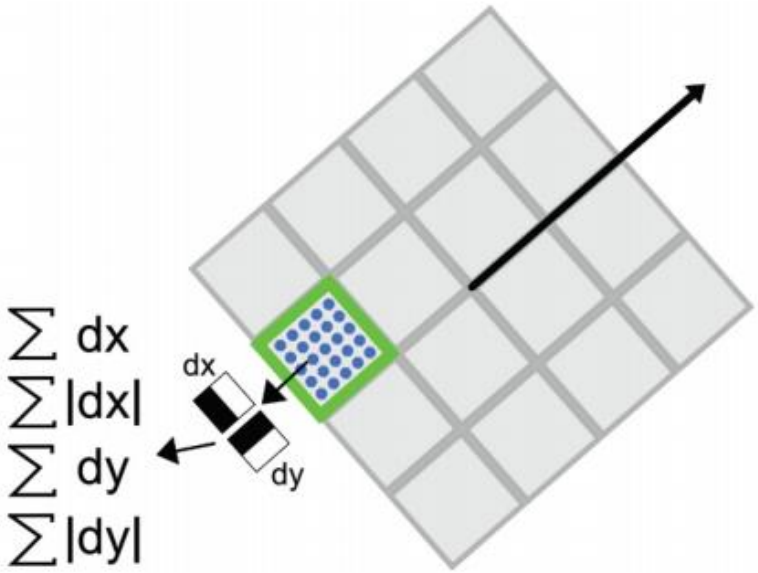
Obrázek 5.6 Princip přiřazování orientací

5.4 Konstrukce deskriptoru

Jak je uvedeno v [25], deskriptor je struktura popisující vlastnosti bodu, které daný bod charakterizují. Hraje tak hlavní roly při porovnávání obrázků a jejich kvalit ovlivňuje náchylnost na různá geometrická zkreslení obrazu. K jejich generování je využito okolí významného bodu.

Podle [26], při konstrukci deskriptoru se nejprve vymezuje čtvercová oblast se středem v místě významného bodu a zároveň je tato oblast natočena podle dominantního vektoru, viz kapitola 5.3. Tato čtvercová plocha je rozdělena na 16 stejně velkých čtvercových oblastí a v každé této oblasti se nachází 25 vzorkovacích bodů. Na každý bod se aplikují Haarovo vlnkové filtry, stejně jako v předchozí kapitole 5.3. Každá z těchto 16 oblastí je popsána pomocí 4 rozměrného vektoru, viz vzorec (12), kde d_x a d_y jsou výsledné hodnoty získané pomocí Haarovo vlnkového filtru pro osu x a y u čtvercové oblasti natočené podle dominantního úhlu. Výsledný vektor obsahuje sumu všech výsledných hodnot pro danou oblast a jejich absolutní hodnoty. Výsledný deskriptor je tak popsán pomocí 64 hodnot, které se nazývají biny.

$$v = [\,\sum d_x, \sum d_y, |\sum d_x|, |\sum d_x|\,] \tag{12}$$



Obrázek 5.7 Ilustrace postupu výpočtu SURF deskriptoru [26]

5.5 Porovnávání deskriptorů

Pro porovnávání dvou deskriptorů se počítá jejich euklidovská vzdálenost. S klesající vzdáleností jsou si body podobnější. Euklidovská vzdálenost dvou deskriptorů se vypočte podle vztahu (13).

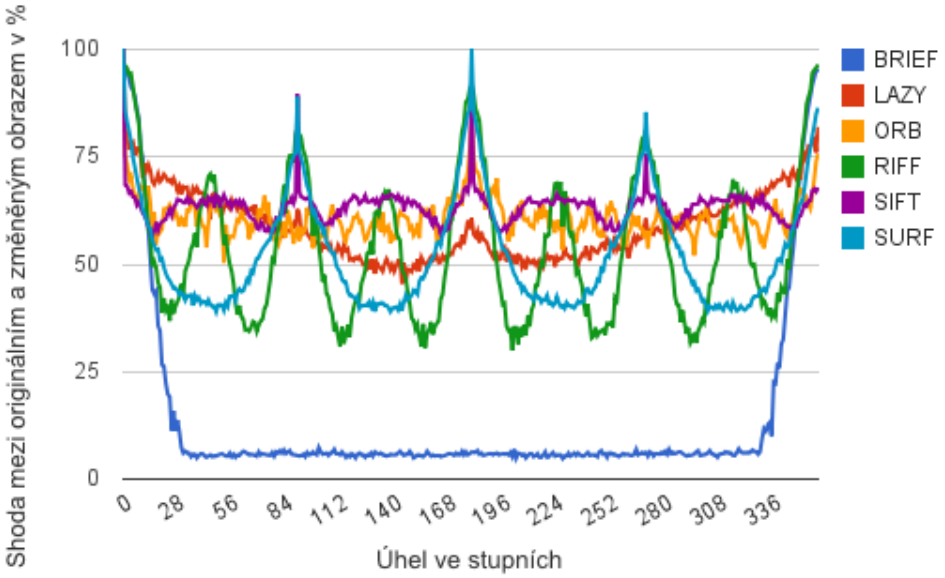
$$d = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2} \tag{13}$$

Kde hodnoty $a_1, a_2, \ldots, a_n$ jsou biny prvního deskriptoru, hodnoty $b_1, b_2, \ldots, b_n$ jsou biny druhého deskriptoru a *n* představuje počet binů. U SURF algoritmu je binů 64.

5.6 Porovnání přesnosti detektorů

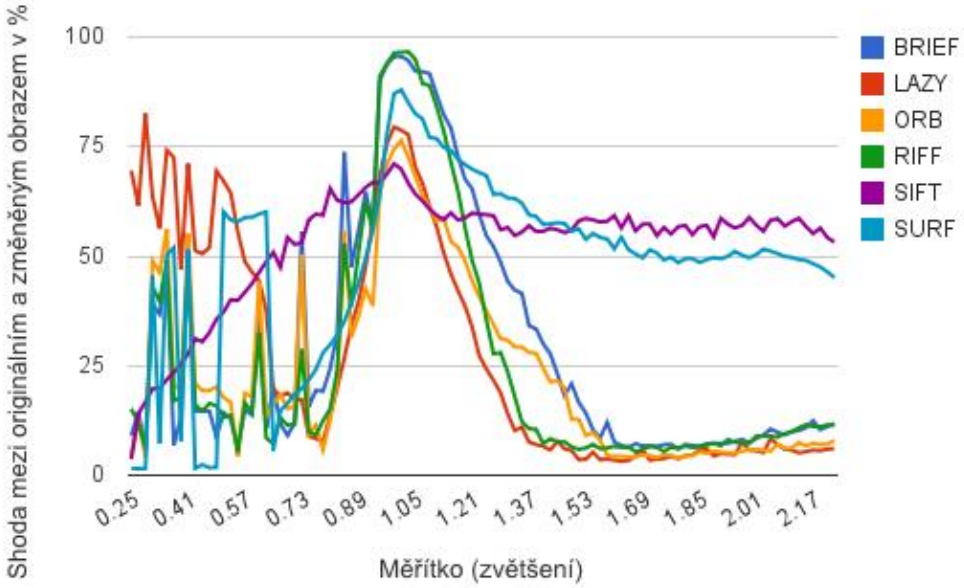
Jak bylo zmíněno v úvodu, pro detekci významných bodů a výpočtu jejich deskriptorů existuje celá řada metod. Tato kapitola se zabývá porovnáním některých metod v závislosti na změně otočení a velikosti obrazu. Porovnávány jsou algoritmy BRIEF, LAZY, ORB, RIFT, SIRF a SURF.

Graf na obrázku 5.8 ukazuje, jak jsou závislé deskriptory na změnu orientace. Osa *x* popisuje natočení deskriptoru a osa *y* míru shody. Z grafu vyplývá, že všechny algoritmy kromě BRIEF jsou invariantní vůči natočení. Změna stability mezi ostatními algoritmy při změně natočení, je dána různými algoritmy pro výpočet deskriptorů.



Obrázek 5.8. Závislost přesnosti na změně otočení [27]

Graf na obrázku 5.9 ukazuje, jak jsou závislé deskriptory na změnu velikosti. Osa *x* popisuje změnu velikosti deskriptoru a osa *y* míru shody. Nejlepší stabilitu vykazují algoritmy SIRF a SURF protože počítají s deskriptory o různých velikostech. U ostatních algoritmů je velikost deskriptorů neměnná.



Obrázek 5.9 Závislost přesnosti na změně velikosti [27]

6 Návrh řešení

Cílem této kapitoly je navrhnout základní funkcionality administrátorské, serverové a klientské aplikace. Jednotlivé funkcionality, které budou zahrnuty do výsledných aplikací, jsou popsány jak textovou formou, tak i pomocí Use Case diagramů. Kapitola se dále zabývá návrhem databáze a příslušných tabulek, kterou bude serverová aplikace využívat.

Architektura aplikace bude postavena na síťovém modelu klient-server. K dispozici budou dvě klientské aplikace. Jedna aplikace je určena pro počítače s operačním systémem Windows a druhá aplikace je určena pro mobilní zařízení s operačním systémem Android. U obou aplikací je kladen důraz na jednoduchost pro co nejjednodušší ovládání. Pro účely vyhledávání se tyto klientské aplikace připojují na server, na kterém běží aplikace pro vyhledávání. Klientská aplikace pošle pořízený obraz, který se nazývá vzor, serverové aplikaci. Ta začne daný vzor vyhledávat ve své databázi obrazů. Pro správu obrazů slouží administrátorská aplikace, která umožňuje obrazy přidávat, upravovat nebo mazat. Přidávání obrazů je možné z lokálního disku nebo z externího zdroje pomocí URL adresy. Data se přenáší mezi klientskou a serverovou aplikací pomocí několika objektů. Ty jsou realizovány pomocí knihovny, která rozšiřuje všechny aplikace.

6.1 Použité technologie a nástroje

Pro psaní aplikace byl zvolen programovací jazyk Java, jehož hlavní výhodou je snadný vývoj a přenositelnost. Aplikace lze spouštět na zařízeních či počítačích, které mají k dispozici interpret Javy, který se nazývá Java Virtual Machine (JVM).

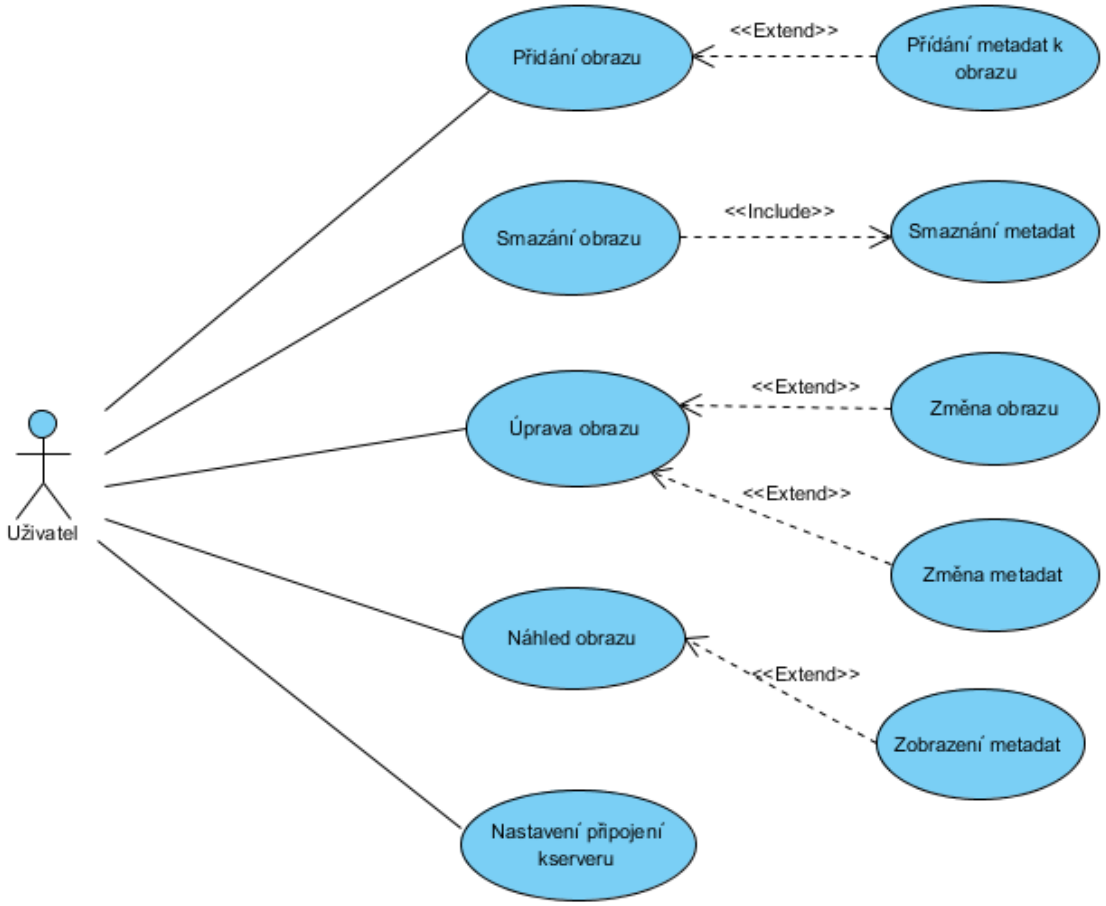
Jako pomocná knihovna pro práci s obrazem byla zvolena knihovna OpenCV (Open source computer vision). Tato knihovna je pod licencí BSD, a proto je ji možné používat zdarma pro akademické i komerční použití. Je také dostupná pro nejrůznější programovací jazyky jako jsou C, C++, Python a od roku 2013 je tato knihovna dostupná i pro Javu. Pro vývoj byla zvolena knihovna verze 2.4.3.

Pro uchovávání metadat je zvolena dvojice relačních databází, a to MySQL a PostreSQL, mezi kterými si může uživatel vybrat. Tyto databázové systémy byly zvoleny, protože jsou volně šiřitelné a hlavně jsou dostupné jak pro operační systém Windows, tak i pro operační systém Linux.

Pro vývoj desktopových aplikací je zvoleno IDE NetBeans, které je určeno hlavně pro vývoj aplikací v Javě. Mobilní aplikace je vyvíjena pro operační systém Android a pro její vývoj je použito IDE Eclipse, které je lépe vybaveno pro psaní mobilních aplikací než IDE NetBeans.

6.2 Návrh administrátorské aplikace

Administrátorská aplikace slouží pro správu obrazů v databázi. Princip fungování je popsán Use Case diagramem na obrázku 6.1. Aplikace umožňuje přidávání obrazů, mazání, upravování a náhled. Při vyhledávání dochází k předzpracování obrazu a výsledná data se používají při porovnávání se vzorem. Předzpracovaná data z obrazu nejsou závislá na hledaném vzoru a tak jsou tyto data vždy stejná pro daný obraz. Proces předzpracování zabírá nemalý výpočetní čas, a tak je vhodné tento krok provádět již při samotném vkládání obrazu v administrátorské aplikaci. Předzpracovaná data si poté serverová aplikace pouze načte, což je mnohem rychlejší a dochází tak k podstatnému urychlení vyhledávání. Aplikace dále umožňuje nastavení připojení k serveru.



Obrázek 6.1 Use Case diagram administrátorské aplikace

6.3 Návrh serverové aplikace

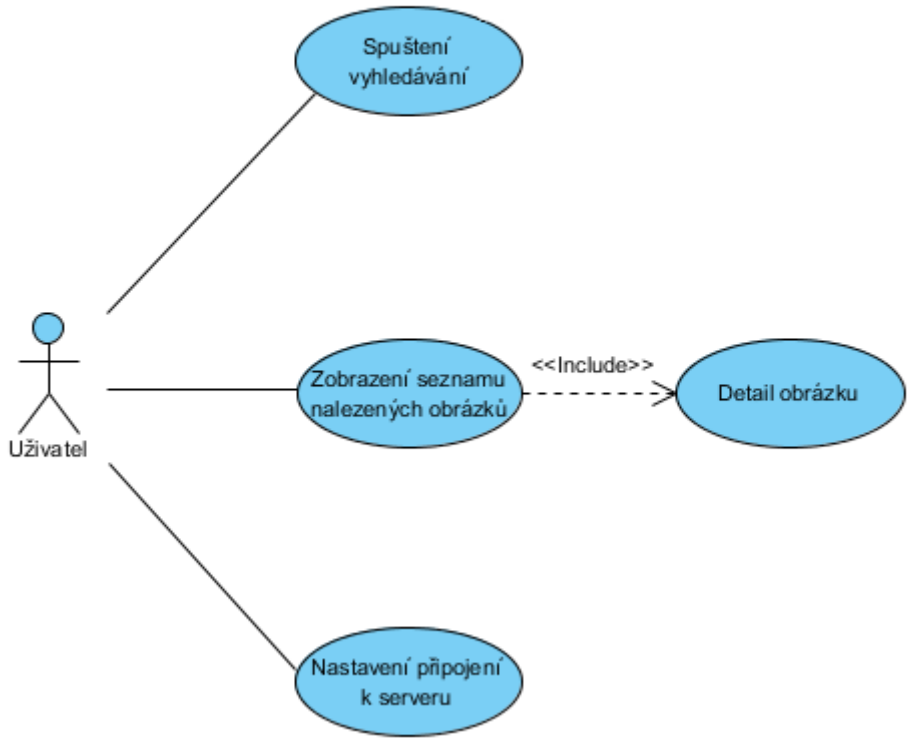
Serverová aplikace se používá pro vyhledávání obrazů v databázi. Fungování aplikace popisuje Use Case diagram na obrázku 6.2. Při spuštění aplikace dojde k načtení obrazů do paměti aplikace pro její rychlejší fungování. Po připojení klienta obdrží serverová aplikace hledaný vzor, který se bude porovnávat s načtenými obrazy pomocí předem vybraného

algoritmu. Aplikace v průběhu vyhledávání posílá mezivýsledky ke klientovi. Serverová aplikace bude zobrazovat seznam právě běžících vyhledávacích procesů, u kterých jsou zobrazeny informace o připojeném klientovi, jako je IP adresa, čas spuštění vyhledávání apod. Dále aplikace bude zobrazovat prostřednictvím konzole nejrůznější stavové informace o serveru a o připojených klientech. V nastavení aplikace bude možné nastavit počet vláken pro vyhledávání, které slouží k paralelizaci, vybrání algoritmu, který bude použit pro vyhledávání, nastavení připojení k databázi a nastavení složky, ve které budou obrazy uloženy.

Obrázek 6.2 Use Case diagram serverová aplikace

6.4 Návrh klientských aplikací

Klientské aplikace budou navrženy pro co nejjednodušší ovládání a tak i snadné použití. Fungování aplikace popisuje Use Case diagram na obrázku 6.3. Po připojení k serveru se odešle vzor, který se má vyhledávat. Server bude postupně zasílat mezivýsledky, které se budou zobrazovat na klientské aplikaci. Uživatel si během vyhledávání tyto získané mezivýsledky může prohlížet a v případě nalezeného obrazu může vyhledávání předčasně ukončit. Vzor pro vyhledávání lze získat u mobilního zařízení s operačním systémem Android pomocí fotoaparátu nebo lze načíst již uložené obrázky jako vzor z „Galerie“ daného zařízení. V případě aplikace pro počítače lze hledaný vzor načíst z lokálního umístění na disku. Aplikace disponují jednoduchým nastavením, které slouží k připojení k serveru.



Obrázek 6.3 Use Case diagram klientských aplikací

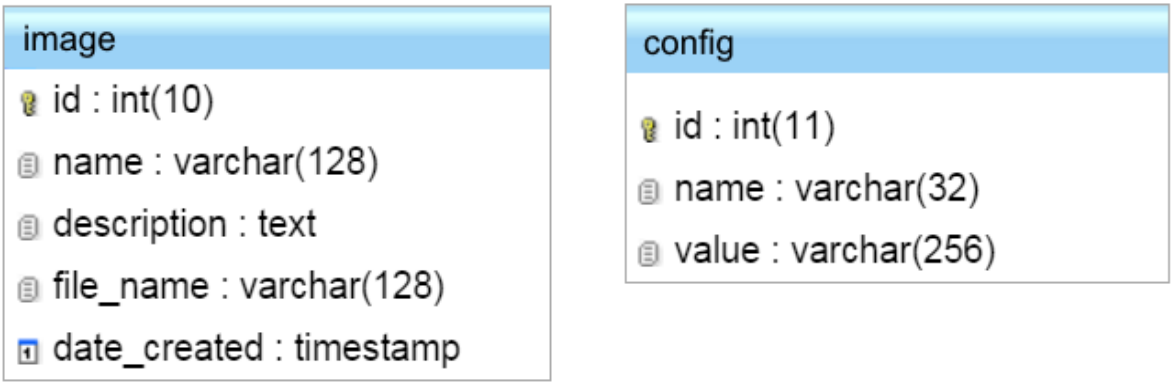
6.5 Návrh databáze

Databáze se skládá z tabulky „image“ a tabulky „config“. Tabulka „image“ se používá pro ukládání metadat o obrazech a má 5 sloupců:

- id – slouží jako unikátní identifikátor
- name – slouží pro uchovávání názvu obrazu
- description – slouží pro uchovávání popisu obrazu
- file_name – slouží pro uložení názvu souboru
- data_created – obsahuje datum vytvoření záznamu a vložení obrazu

Druhá tabulka „config“ je použita pro uložení konfiguračních dat a má 3 sloupce.

- id – slouží jako unikátní identifikátor
- name – slouží k nadefinování názvu konfigurační proměnné
- value – slouží k nadefinování hodnoty pro konfigurační proměnnou

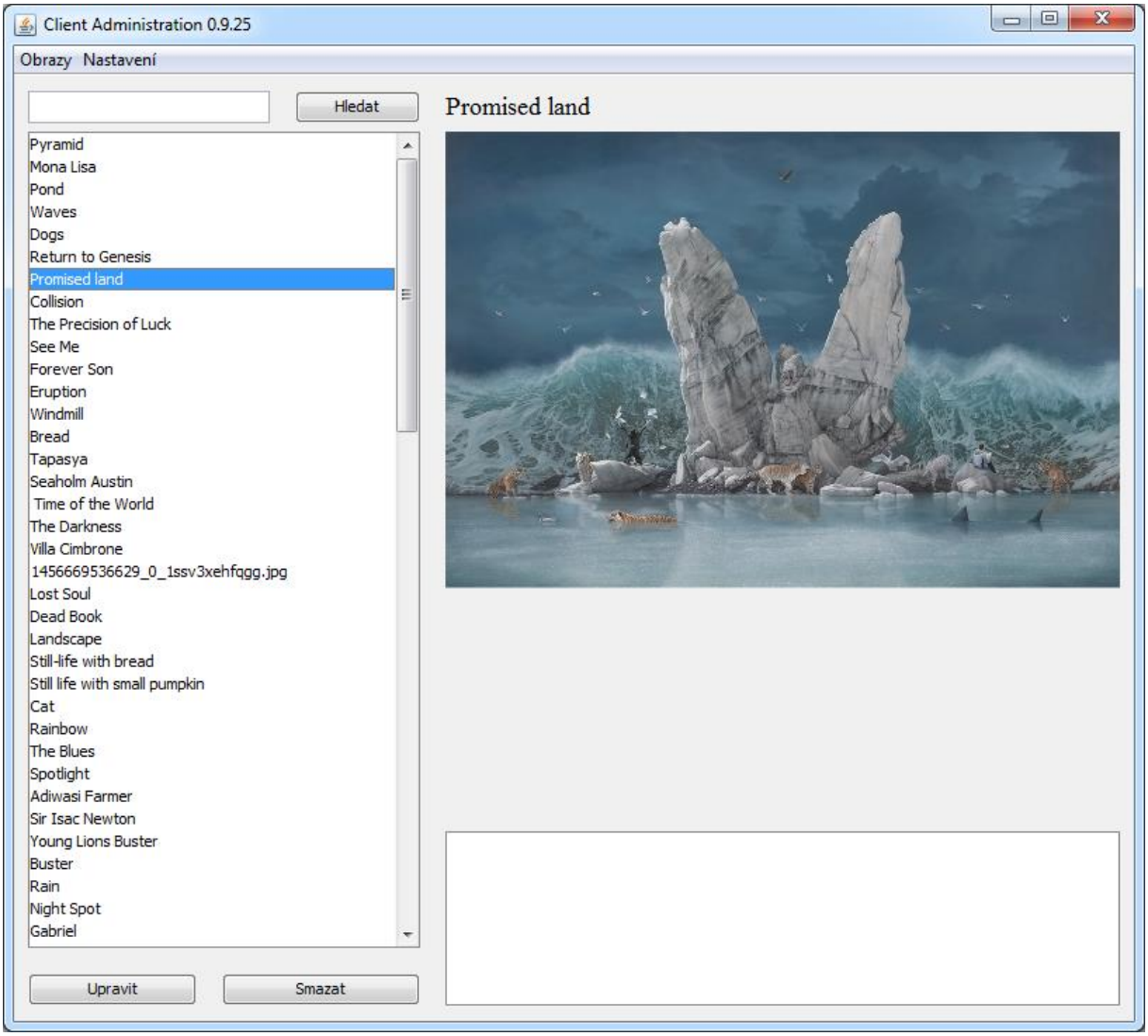


Obrázek 6.4 Návrh tabulek pro databázi

7 Administrátorská aplikace

Tato aplikace slouží pro správu obrazů, ve kterých se vyhledává. Při jejich ukládání dochází k předzpracování dat k obrazům, které jsou použity při vyhledávání.

Aplikace má jednoduché grafické rozhraní, které lze rozdělit do několika logických částí. V horní části se nachází menu s dvěma položkami „Obrazy“ a „Nastavení“. Pomocí položky „Obrazy“ lze přidávat nové obrazy do databáze. Ty lze vybírat z umístění na lokálním disku, nebo z externího zdroje za pomoci URL adresy. Druhá položka „Nastavení“ umožňuje nastavení připojení k serveru. V levé části se zobrazují všechny uložené obrazy, které lze upravovat a mazat pomocí dvojice tlačítek v levém dolním rohu. V prázdném prostoru, v pravé části aplikace, se zobrazuje vybraný obraz ze seznamu, a pod tímto obrazem jsou vypsána příslušná metadata.



Obrázek 7.1 Grafické uživatelské rozhraní administrátorské aplikace

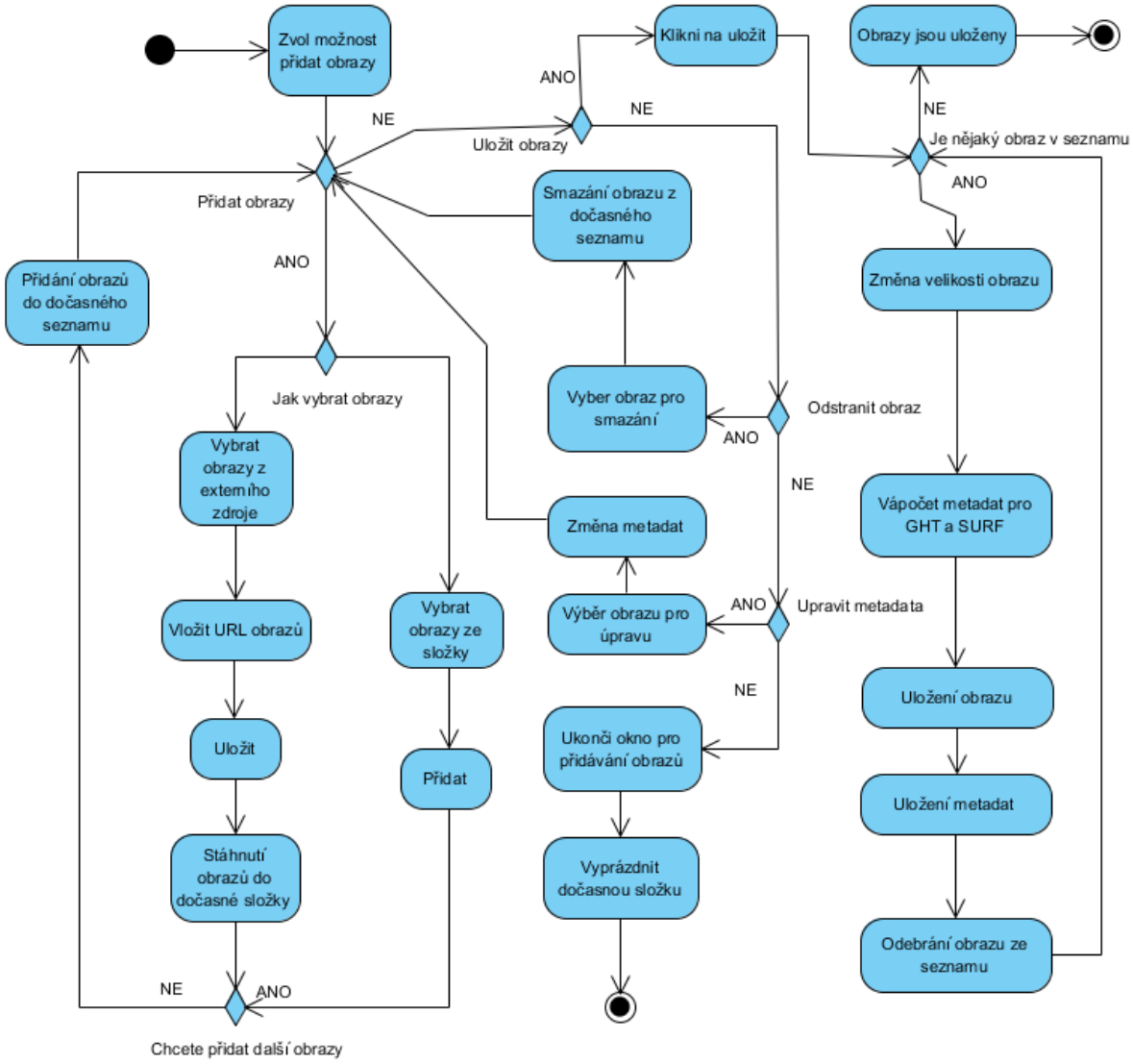
7.1 Popis funkcionalit aplikace

Hlavní funkcionalitou aplikace je správa obrazů, která je znázorněna aktivity diagramem na obrázku 7.2. Při spuštění se aplikace pokusí jako první navázat spojení se serverovou aplikací. Pokud se spojení nepodaří navázat, aplikace informuje uživatele příslušnou chybovou hláškou. Při úspěšném navázání spojení obdrží aplikace seznam všech obrazů, které se zobrazí v seznamu v levé části aplikace. Posílají se ovšem jen názvy obrazů, samotné obrazy se neodesílají, protože pokud by aplikace obsahovala stovky obrazů, jejich načtení by mohlo trvat velice dlouho. Až po výběru konkrétního obrazu ze seznamu, aplikace požádá server o samotný obraz a jeho metadata. Načtení jednoho obrazu trvá velice krátkou dobu a uživatel si nemusí ani všimnout, že je obraz načítán až nyní.

Po výběru možnosti pro přidání obrazu je otevřeno nové okno, ve kterém se nachází tzv. dočasný seznam. Po vybrání obrazů jsou do tohoto dočasného seznamu uloženy reference na vybrané obrazy, a právě zde je možné k obrazům přidávat metadata. Pokud jsou vybrány obrazy z externího zdroje pomocí URL adresy, jsou tyto obrazy nejprve staženy do dočasné složky „temp“ nacházející se v kořenovém adresáři aplikace a poté je do dočasného seznamu uložena reference na ně. Velikost všech obrazů je změněna, kdy nejdelší strana obrazu je 480px. Všechny obrazy v databázi mají tedy nejdelší velikosti strany 480px. Po zvolení možnosti „Uložit“ jsou postupně obrazy odesílány na server, kde probíhá jejich předzpracování a předpříprava dat, které budou použity pro vyhledávání. Aby bylo možné obrazy odeslat, musí být převedeny do pole bytů. Na straně serveru poté dochází k opětovné rekonstrukci obrazu z tohoto pole. Po uložení všech obrazů do databáze je poté dočasná složka vymazána.

Pro úpravu obrazu lze použít tlačítko „Upravit“ v dolní části obrazovky nebo lze dvakrát poklepat na název obrazu ze seznamu. Poté je otevřeno jednoduché okno pro úpravu obrazu a jeho metadat. Lze měnit všechna metadata o obrazu a i obraz samotný. Ten lze načíst, stejně jako při přidávání obrazu, z lokálního umístění, nebo z externího zdroje pomocí URL. Po stisknutí tlačítka „Uložit“ jsou pro uložení odeslány na server pouze změněná data. Pokud je vložen nový obrázek, dochází také k předpřípravě dat pro vyhledávání.

Při jakékoliv manipulaci s obrazy, jako je přidávání, upravování a mazání, dochází okamžitě i k úpravě již načteného seznamu obrazů v serverové aplikaci. Serverová aplikace se tedy nemusí restartovat pro načtení provedených změn.



Obrázek 7.2 Aktivity diagram administrátorské aplikace

7.1.1 Předpříprava dat pro obecnou Houghovu transformaci

Pro získání předzpracovaných dat, které by byly využity u obecné Houghovy transformace se na obraz nejprve aplikuje eroze, která je popsána v kapitole 3.2.1, o velikosti masky 5x5, díky které dojde k odstranění detailů v obraze, které by mohly způsobovat nežádoucí zkreslení. Lze ji nahradit i mírným rozmazáním obrazu. Eroze ovšem vrací nejlepší výsledky pro účely této aplikace, viz obrázky 7.3, 7.4, 7.5 a 7.6. Druhým krokem je poté aplikování Cannyho hranového detektoru, viz kapitola 3.3.3. Výsledkem jsou tak černobílé obrazy, kdy jsou bílou barvou znázorněny hrany.

Obrázek 7.3 Originální obraz

Obrázek 7.4 Obraz pouze po aplikování detektoru hran

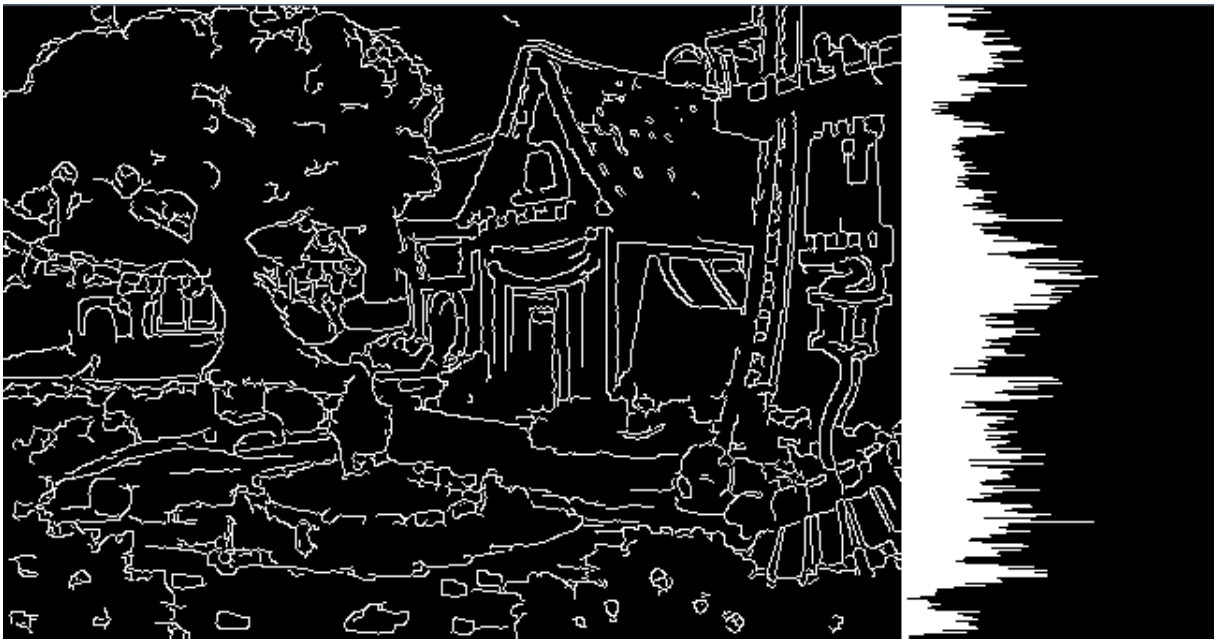
Obrázek 7.5 Obraz po rozmazání a aplikace detektoru hran

Obrázek 7.6 Obraz po aplikování Eroze a detektoru hran

Při prohledávání obrazu se musí projít přes všechny jeho body, jak černé tak i bílé. Při porovnávání se ovšem počítá pouze s bílými body a zbytečné procházení černých bodů způsobuje další zpomalení. Proto je vhodné uložit všechny pozice bílých bodů, se kterými se bude počítat. Algoritmus tak může procházet pouze tyto bílé body.

Důležité je také, jakým způsobem se tyto bílé body uloží. Ve většině případů bude hledaný vzor pouze z části obrazu, nikoli z celého, a tak pokud je vzor umístěn do levého horního rohu, je zbytečné pracovat s bílými body obrazu z pravého dolního rohu. Je tedy vhodné, aby se počítalo pouze s body v obraze, které leží v prostoru vzoru. Pokud by byl seznam bílých bodů uložen jen do jednorozměrného pole, nebylo by možné efektivně vyselektovat body, které se nachází v prostoru vzoru. Proto jsou bílé body z obrazu uloženy do dvourozměrného pole, jehož struktura je znázorněna na obrázku 7.7. Každý řádek z obrazu má vlastní pole, ve kterém se nachází bílé body z daného řádku. Z této struktury je tak možné snáze vybrat body,

které odpovídají pozici vzoru v obraze. Podrobný popis využití tohoto dvourozměrného pole je popsán v kapitole 8.1.1.



Obrázek 7.7 Grafická reprezentace dvourozměrného pole pro bílé body z obrázku vlevo

Pro každý černobílý obraz je také vypočten jeho integrální obraz, viz kapitola 5.1. Ten je využit před porovnáváním vzoru s obrazem pomocí obecné Houghovy transformace. Nejprve dochází k porovnání počtu bílých bodů vzoru s oblastí obrazu, kde se vzor nachází. Pokud je černá barva v obraze reprezentována hodnotou 0 a bílá barva hodnotou 1, pak lze pomocí integrálního obrazu zjistit počet bílých bodů v dané oblasti. K využití obecné Houghovy transformace dochází jen v případě, kdy je počet bílých bodů podobný jak u vzoru, tak i ve vybrané oblasti obrazu.

7.1.2 Předpříprava dat pro SURF algoritmus

Pro předpřípravu dat pro SURF algoritmus je nejprve nutné převést zvolený obraz do odstínu šedi, protože SURF algoritmus pracuje pouze s těmito obrazy. Dalším krokem je detekce významných bodů v obrazu. To je provedeno podle postupu, který je uveden v kapitole 5.2. Ukázka detekovaných významných bodů v obraze je znázorněna na obrázku 7.8. Posledním krokem je výpočet deskriptorů pro všechny nalezené významné body, pomocí kterých dochází k porovnávání obrazů se vzorem. To je provedeno podle kapitoly 5.4. Detekované významné body a jejich deskriptory jsou poté uloženy spolu s obrazem. Pro převod obrazu do odstínu šedi, nalezení významných bodů v obrazu a vypočítání jejich deskriptorů je využita knihovna OpenCV.

Obrázek 7.8 Ukázka obrazu, ve kterém jsou vyznačeny nalezené významné body

7.2 Popis tříd

Tato aplikace je složena z trojice balíčků, kterými jsou administration.core, administration.gui a administration.gui.img. Balíček administration.core obsahuje třídy realizující hlavní logiku aplikace. Balíček administration.core obsahuje třídy, které realizují především grafické uživatelské rozhraní. Poslední balíček adminstration.gui.img obsahuje sadu ikon, které jsou v aplikaci zobrazovány. Class diagramy jsou součástí přílohy.

7.2.1 Balíček administration.core

- AdministrationClient – Hlavní třída, která obsahuje metodu main. V této metodě dochází k inicializaci OpenCV knihovny, grafického uživatelského rozhraní a načtení konfiguračního objektu Config, který se uloží jako statická proměnná.

- Client – Třída obsahuje sadu metod pro komunikaci se serverovou aplikací. Pomocí této třídy dochází k odesílání, ale i čtení dat. Všechna komunikace probíhá přes instanci třídy Socket. Samotná data jsou odesílána jako serializované objekty. Metody z této třídy jsou použity tak, že jsou volány jinými metodami, které jsou implementovány v grafickém uživatelském rozhraní.

- Config – Třída obsahuje veškeré konfigurační vlastnosti aplikace. Mezi ně patří nastavení připojení k serveru, název dočasné složky pro ukládání obrazů a maximální velikost obrazů.

- Image – Třída realizuje datový objekt. Každá instance této třídy reprezentuje právě jeden obraz nesoucí jeho metadata.

- UpdateStatus – Pomocná třída, která je využita při úpravě obrazů. Obsahuje informace o tom, které části obrazu byly změněny. Podle toho jsou na server poté odeslána pouze změněná data.

- UtilsL – Pomocná třída, která obsahuje metodu pro vykreslení obrazu z objektu typu Mat a metodu pro převod objektu Mat do pole bytů.

7.2.2 Balíček administration.gui

- MainWindow – Hlavní grafické uživatelské rozhraní rozšířené třídou JFrame. Zobrazuje se zde seznam vložených obrazů, detail konkrétního obrazu, dvojice tlačítek pro editaci či mazání obrazů a hlavní menu. Z menu lze vybrat možnosti pro přidávání obrazů či nastavení připojení k serveru.

- DrawPanel – Panel pro vykreslování obrazů z objektu Mat, který je rozšířen třídou JPanel.

- ImageAddWindow – Třída sloužící pro vybrání obrazů, která je rozšířená třídou JFrame. Umožňuje vybrání obrazů z pevného disku, nebo z URL adresy. Vybrané obrazy jsou v této třídě zobrazeny a je možné jim přiřadit medatada. Obrazy zobrazené v tomto grafickém rozhraní nejsou uloženy. K jejich uložení dochází až po stisknutí tlačítka „Uložit“.

- ImageEditWindow – Třída je rozšířena třídou JFrame a slouží pro editaci již vložených obrazů.

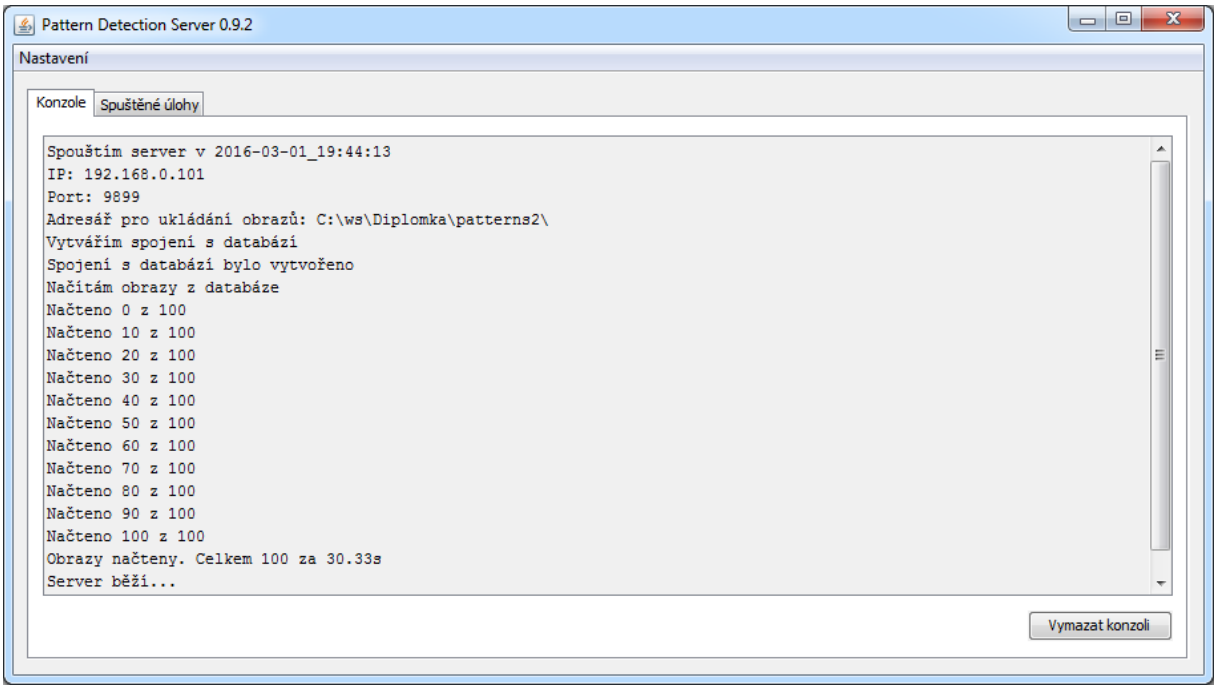
- ImageUrlAddWindow – Jednoduchá třída rozšířena třídou JFrame. Osahuje pole, do kterého se vloží URL adresy k obrazům. Tyto obrazy jsou poté staženy do dočasné složky „temp“. Po uložení obrazů dochází k smazání obsahu této složky.

- SettingsServerWindow – Třída je rozšířena třídou JFrame, která slouží k nastavení připojení k serveru. Obsahuje dvě pole pro IP adresu a port serveru.

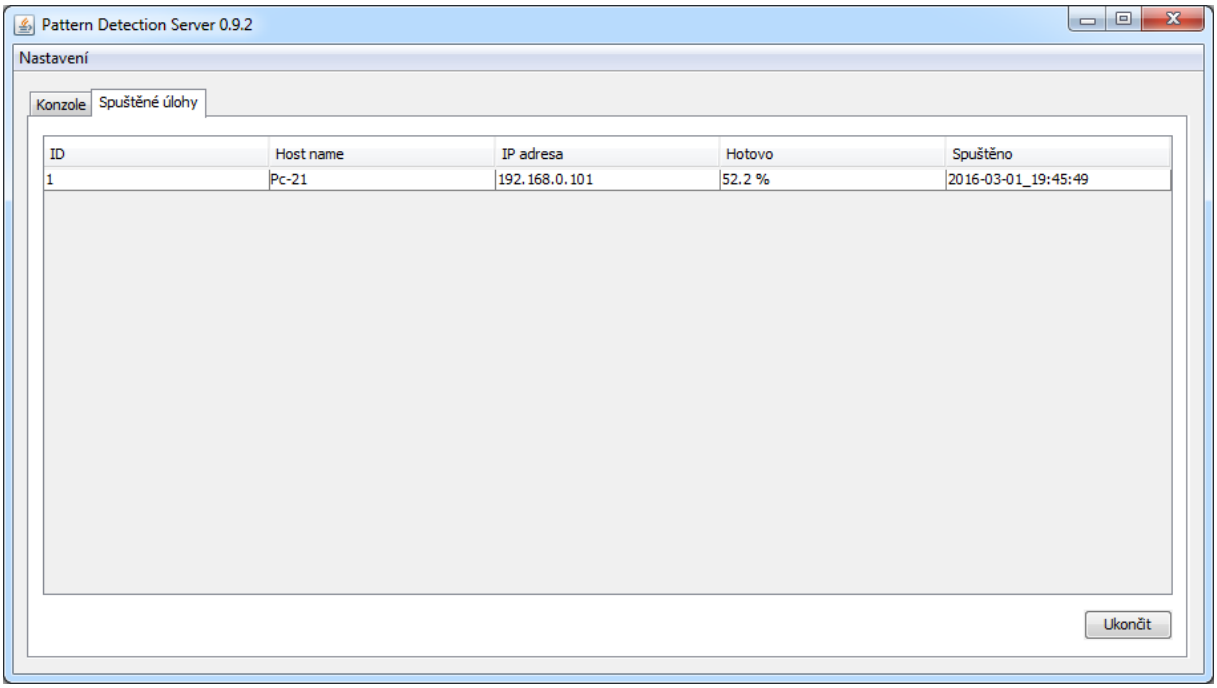
8 Serverová aplikace

Tato aplikace se stará o samotné vyhledávání. K tomu lze využít jednu ze dvou metod, kterými jsou obecná Houghova transformace a SURF algoritmus. Popsání funkcionalit obecné Houghovy transformace v této aplikaci je velice obsáhlé, protože je vytvořena od úplného základu, bez použití jakékoliv knihovny. Je zde popsáno i několik optimalizačních technik, které jsou při vyhledávání využity. Oproti tomu, SURF algoritmus je implementován pomocí OpenCV knihovny, a tak popis funkcionalit není tak rozsáhlý. Na konci této kapitoly se nachází popis všech tříd, které jsou v aplikaci použity.

Serverová aplikace má jednoduché grafické uživatelské rozhraní, které se skládá z hlavního menu a dvou záložek. Hlavní menu obsahuje položku „Nastavení“, pomocí které je možné nastavit připojení k databázi, nastavit vyhledávací algoritmus, počet vláken na kterých bude výpočet probíhat a nastavení složky pro ukládání obrazů. Záložka „Konzole“ slouží pro zobrazení událostí, které mohou při běhu nastat, jako je informace o připojení k databázi, připojení klienta, odpojení klienta atd. viz obrázek 8.1. Druhá záložka „Spuštěné úlohy“ zobrazuje informace o právě probíhaném hledání, viz obrázek 8.2.



Obrázek 8.1 GUI serverové aplikace - Konzole

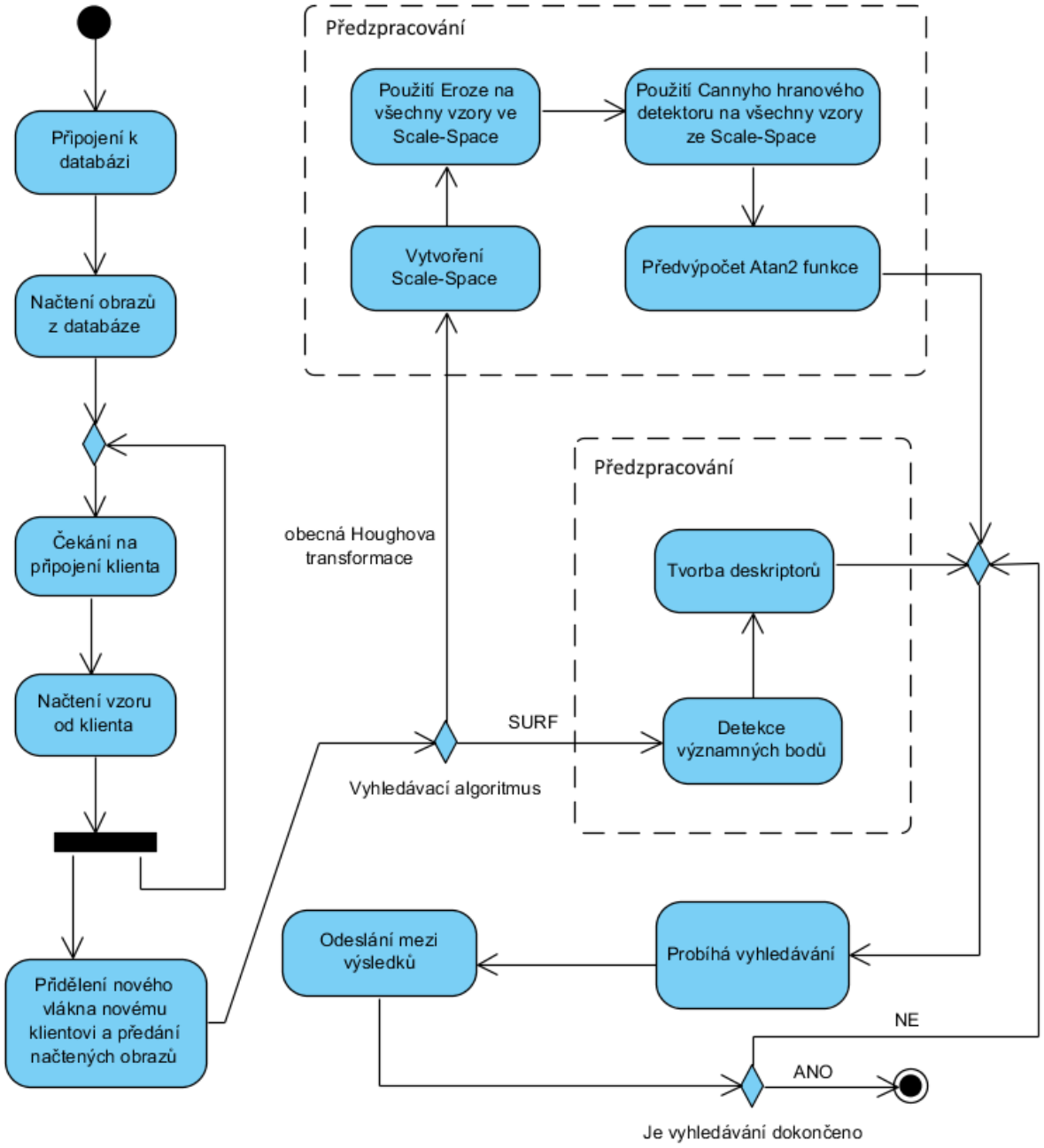


Obrázek 8.2 GUI serverové aplikace - Spuštěné úlohy

8.1 Popis funkcionalit aplikace

Hlavní princip fungování aplikace je znázorněn aktivity diagramem na obrázku 8.3. Při spuštění se aplikace nejprve pokusí připojit k databázi. Po jejím připojení jsou načteny metadata o obrazech z databáze, jako je název obrazu, popis a název souboru ve kterém jsou uloženy předzpracovaná data, které jsou následně také načtena. Předzpracovaná data se načítají ze serializovaného objektu typu DataImageStore. Všechny informace o obrazech se načítají do objektů, které jsou typu Image. Aplikace je nyní připravena a čeká na připojení klienta. Důvodem pro načtení obrazů při startu aplikace je to, že načítání může trvat i desítky sekund v závislosti na množství načítaných obrazů a každý klient by tak musel vždy čekat dlouhou dobu, než se obrazy načtou. Proto jsou načtené obrazy pouze překopírovány připojenému klientovi, což je operace, která zabere pouze okamžik.

Klient a server komunikují prostřednictvím třídy Socket. Klientovi je po připojení vyhrazeno nové vlákno, což umožňuje připojení více klientů zároveň. Jak bylo zmíněno výše, do tohoto vlákna jsou poté překopírovány již dříve načtené obrazy. Jako první klient posílá vzor, který se má hledat. Tento vzor je posílán jako pole bytů, ze kterého je poté na straně serveru opět vytvořen obraz. Vzor se poté ukládá do objektu typu Mat, který využívá OpenCV knihovna pro práci s obrazy. Dále dochází k ověření velikosti vzoru. Ten by měl mít nejdelší stranu dlouhou 300px. V případě, že tomu tak není, dochází ke změně velikosti na tuto hodnotu. Velikost 300px byla zvolena, aby bylo především sníženo množství přenášených dat od klienta k serveru. Velikost může být tedy i větší, ale ne více než 480px, což je největší velikost obrazu ve kterém se vyhledává, protože se předpokládá, že hledaný vzor bude maximálně stejně velký, jako obraz. Další kroky jsou závislé na zvoleném druhu vyhledávácího algoritmu.



Obrázek 8.3 Aktivity diagram serverové aplikace

8.1.1 Obecná Houghova transformace

Při zvolení obecné Houghovy transformace pro vyhledávání je dalším krokem postupné zvětšování a zmenšování vzoru. Tímto postupem tak vzniká tzv. Scale-Space, který obsahuje množinu vzorů o různých velikostech a je tak zajištěna invariance vůči velikosti. Krok pro změnu velikostí je stanoven na 15px. Tato hodnota je získána na základě testů, jejichž výsledek poskytuje poměrně velké zrychlení za cenu nepřesnosti, která je ještě pořád dostatečně malá.

Na vzory je poté aplikována funkce Eroze, která je popsána v kapitole 3.2.1, o velikosti masky 5x5, díky které dojde k odstranění detailů v obraze, které by mohly způsobovat zkreslení. Druhým krokem je poté aplikování Cannyho hranového detektoru, viz obrázek 8.4 a 8.5. Výsledkem jsou tak černobílé vzory, kdy jsou bílou barvou znázorněny hrany. Z takto připravených vzorů jsou poté vytvořeny referenční tabulky, viz kapitola 4.3.

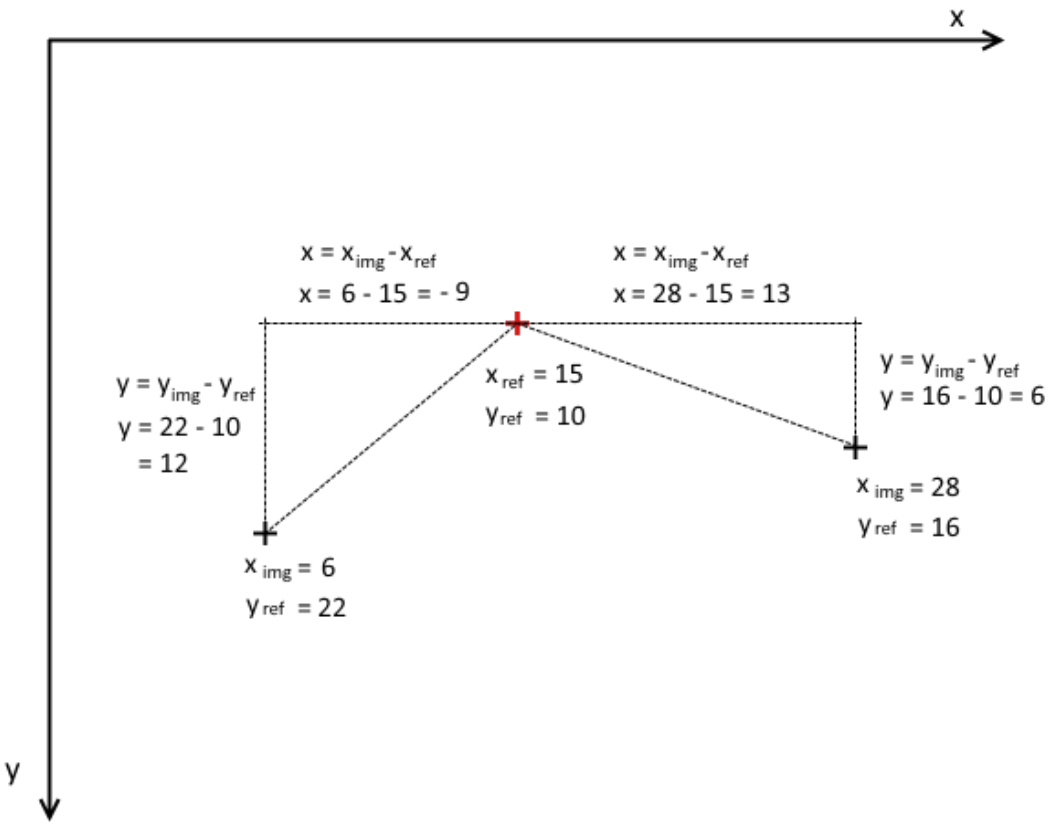
Obrázek 8.4 Originální obrázek

Obrázek 8.5 Obrázek po aplikování Eroze a detektoru hran

Posledním krokem, který se provádí před samotným vyhledáváním je předpočtení hodnot pro metodu java.lang.Math.atan2, která je často používána při vyhledávání. Tato metoda je poměrně výpočetně náročná a významně snižuje rychlost vyhledávání. Proto je vytvořeno dvourozměrné pole, do kterého se předpočtené hodnoty ukládají. Vzdálenosti, které by se použili ve funkci atan2, se zde využívají jako indexy pro přístup k předpočteným hodnotám. Vypočtené hodnoty pomocí funkce atan2 jsou v radiánech a pohybují se v rozmezí od - π do + π. Kvůli dalšímu využití je vhodné převést radiány na stupně a jejich zaokrouhlení na celé stupně a záporné stupně převést na kladné, přičtením hodnoty 360.

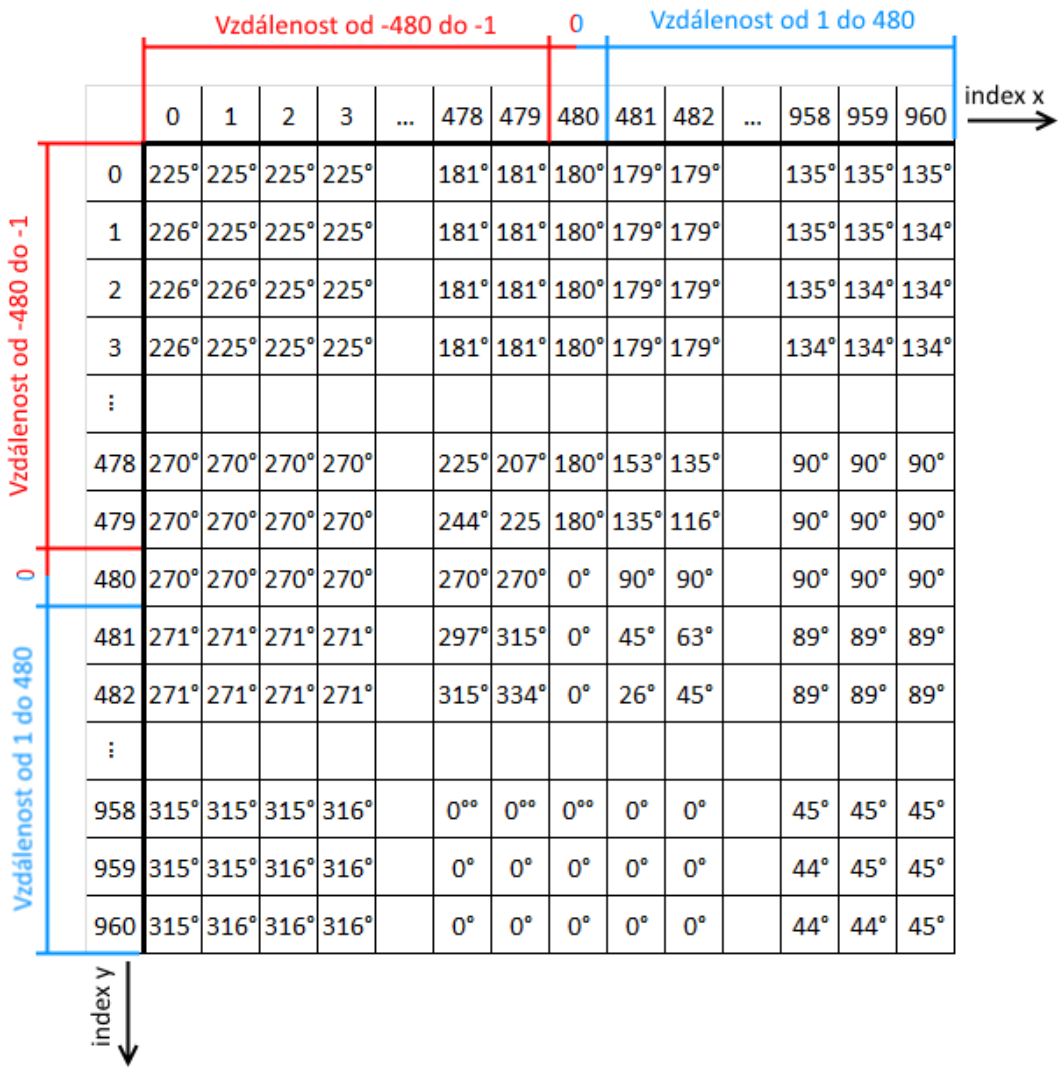
Vzdálenost se vypočte z rozdílu souřadnic referenčního bodu vzoru a daného bílého bodu v obraze. Délka strany *x* se vypočte jako $x = x_{img} - x_{ref}$ kde x_{img} je poloha bílého bodu v obraze a x_{ref} je poloha referenčního bodu vzoru v obraze, pro osu *x*. Toto platí obdobně pro výpočet délky strany pro osu *y*, kdy $y = y_{img} - y_{ref}$.

Pozice bílého bodu v obraze může být před i za referenčním bodem, z čehož vyplývá, že vzdálenost mezi body v ose *x* a *y* může nabývat kladných i záporných hodnot, viz obrázek 8.7. Dále je třeba brát v úvahu, že obraz má nejdelší stranu stanovenou na 480px, viz kapitola 7.1. Vzdálenost tak může v osách *x* a *y* nabývat hodnot od -480 do +480, včetně nuly. Vytvořené dvourozměrné pole má tak rozměry 961, viz obrázek 8.7.



Obrázek 8.6 Znázornění principu počítání vzdáleností mezi referenčním bodem a body v obrazu

Pro přístup do tohoto dvourozměrného pole pomocí vzdálenosti mezi body, kdy vzdálenost slouží jako index, je třeba vždy k dané vzdálenosti přičíst hodnotu 480, protože umístění příslušných hodnot v poli je vždy posunuto o 480. Ukázka části dvourozměrného pole pro předpočtené hodnoty pro funkci atan2 je vidět na obrázku 8.7.



	0	1	2	3	...	478	479	480	481	482	...	958	959	960
0	225°	225°	225°	225°		181°	181°	180°	179°	179°		135°	135°	135°
1	226°	225°	225°	225°		181°	181°	180°	179°	179°		135°	135°	134°
2	226°	226°	225°	225°		181°	181°	180°	179°	179°		135°	134°	134°
3	226°	225°	225°	225°		181°	181°	180°	179°	179°		134°	134°	134°
⋮														
478	270°	270°	270°	270°		225°	207°	180°	153°	135°		90°	90°	90°
479	270°	270°	270°	270°		244°	225	180°	135°	116°		90°	90°	90°
480	270°	270°	270°	270°		270°	270°	0°	90°	90°		90°	90°	90°
481	271°	271°	271°	271°		297°	315°	0°	45°	63°		89°	89°	89°
482	271°	271°	271°	271°		315°	334°	0°	26°	45°		89°	89°	89°
⋮														
958	315°	315°	315°	316°		0°°	0°°	0°°	0°	0°		45°	45°	45°
959	315°	315°	316°	316°		0°	0°	0°	0°	0°		44°	45°	45°
960	315°	316°	316°	316°		0°	0°	0°	0°	0°		44°	44°	45°

Obrázek 8.7 Tabulka reprezentující funkci atan2

Po skončení předzpracování začne samotné vyhledávání vzoru v obrazech, dle kapitoly 4.3. Vzor prochází obraz a na každém místě spočte příslušné hodnoty podle vzorce (7), které jsou uloženy do akumulátoru. Postupně dochází k vyhledávání ve všech obrazech. Po prohledání všech obrazů a použití všech velikostí vzorů, čímž je zajištěna invariance vůči velikosti, dochází k pootočení vzoru a prohledávání obrazů pokračuje pro další úhel natočení. Krok otáčení je stanoven na 15°, jenž poskytuje ještě výsledky s malým zkreslením. Otáčení se ale nemění plynule po 15°. Předpokládá se, že pořízený snímek vzoru, bude ve svislé či vodorovné poloze. Proto jsou jako první prohledávány úhly 0°, 15° a 345° apod. Pořadí úhlů, o které se bude vzor natáčet dle přesně daného pořadí, vypadá následovně: 0°, 15°, 345°, 180°, 195°, 165°, 30°, 330°, 210°, 150°, 315°, 45°, 225°, 135°, 300°, 60°, 120°, 240°, 75°, 285°, 255°, 195°, 90°, 270°. Toto zajišťuje invarianci vůči natočení.

Je velice pravděpodobné, že při vyhledávání je nalezen správný obraz ještě před samotným skončením hledání. Proto jsou uživateli postupně zasílány mezivýsledky a v případě nalezení

správného obrazu je možné, aby sám uživatel ukončil samotné hledání ještě před úplným dokončením.

Samotný princip fungování obecné Houghovy transformace nabízí možnost zrychlení pomocí paralelizace. Její implementace funguje tak, že je přidělen každému vláknu stejný obraz, ve kterém se vyhledávání, ale každému vláknu jsou přiděleny různá místa v obraze, ve kterých se vzor vyhledává, viz obrázek 8.8, na kterém je znázorněn princip rozdělení obrazu pro 4 vlákna. Pro první vlákno jsou vybrány body v obraze 1, 5, 9, 13, 17, 25 a 29. Na těchto pozicích v obraze je postupně umisťován střed vzoru a dochází k vyhledávání. Reálný obraz je samozřejmě mnohem větší než obrázek 8.8. Stejným principem se řídí ostatní vlákna, kterým jsou ovšem přiřazeny jiné pozice v obraze.

Obrázek 8.8 Přidělení pozic v obrazu pro 4 vlákna

Paralelizace není jediná možnost, jak zrychlit běh obecné Houghovy transformace. Další způsob, jak zrychlit vyhledávání je zvětšit krok při prohledávání obrazu. To znamená, že vzor není posouván po obraze po jednom bodu, ale po více. Se zvětšujícím se posunem samozřejmě klesá přesnost. Důvodem proč není při velikosti posunu po obraze 1px shoda 100 % je ta, že se vždy mění velikost vzoru o 15px. Aby byla shoda 100 %, musela by se velikost vzoru měnit po 1px, a také by nesmělo docházet k zaokrouhlování vypočtených úhlů pro referenční tabulku.

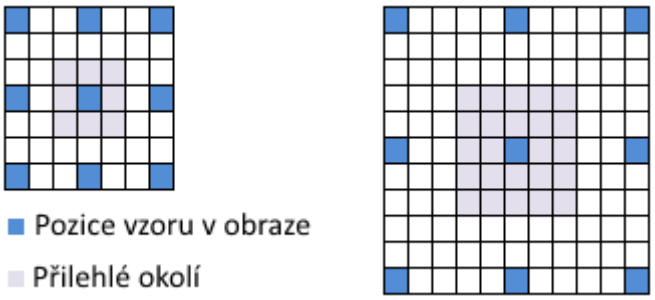
Určitá velikost posunu vzoru po obrazu ještě stačí alespoň pro přibližné určení polohy vzoru. Toho lze využít tak, že po nalezení přibližné polohy vzoru v obraze, je ještě možné prohledat pouze přilehlé okolí tohoto bodu, viz obrázek 8.9. To má za následek zvětšení přesnosti, jako kdyby se obraz procházel po jednom bodu. Přesnost je tak pro každou velikost posunu vzoru po testovaném obraze 82,8 % jak je možné vyčíst z tabulky 8.1. Čím větší velikost posunu vzoru po obraze, tím větší přilehlá oblast musí být prohledávána. Od určité velikosti posunu tak začne docházet ke zpomalení vlivem zvětšující se přilehlé oblasti, která musí být prohledána. Toto je vidět v tabulce 8.1. První sloupec udává velikost posunu vzoru po obraze,

druhý sloupec udává, jak klesá přesnost s rostoucím krokem, třetí sloupec dobu potřebnou pro vyhledávání, bez prohledávání okolí, čtvrtý sloupec dobu vyhledávání včetně prohledávání přilehlého okolí a poslední sloupec velikost prohledávaného okolí.

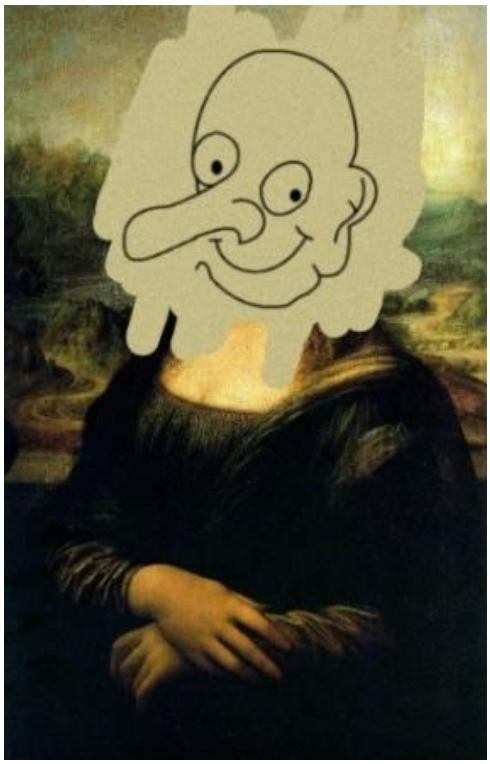
Z tabulky vyplývá, že ideální velikost posunu vzoru po obraze, co se týče do rychlosti, je 7px, protože s větší velikostí posunu začíná docházet ke zpomalení. Na dalších obrázkách 8.10 až 8.13 je možné vidět, jak může vypadat akumulátor pro různé velikosti posunů po obraze a na jakém obrazu a vzoru bylo prováděno toto měření.

Velikost posunu po obraze	Přesnost bez prohledávání přilehlého okolí	Doba vyhledávání bez přilehlého okolí	Doba vyhledávání včetně přilehlého okolí	Velikost přilehlého okolí
1px	82,8 %	9,2 s	9,2 s	0px
2px	80,1 %	2,8 s	2,91 s	9px
3px	74,8 %	1,5 s	1,52 s	9px
4px	74,8 %	0,9 s	1,13 s	25px
5px	71,8 %	0,7 s	0,82 s	25px
6px	70,9 %	0,67 s	0,76 s	49px
7px	63,3 %	0,56 s	0,67 s	49px
8px	65 %	0,43 s	0,8 s	81px
9px	58,3 %	0,41 s	0,79 s	81px
10px	59 %	0,37 s	0,82 s	121px
11px	74,1%	0,34 s	0,84 s	121px

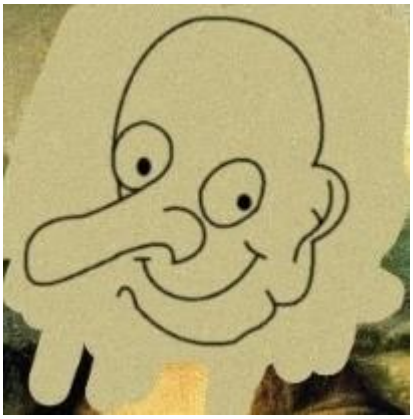
Tabulka 8.1 Tabulka závislostí na velikosti posunu po obraze



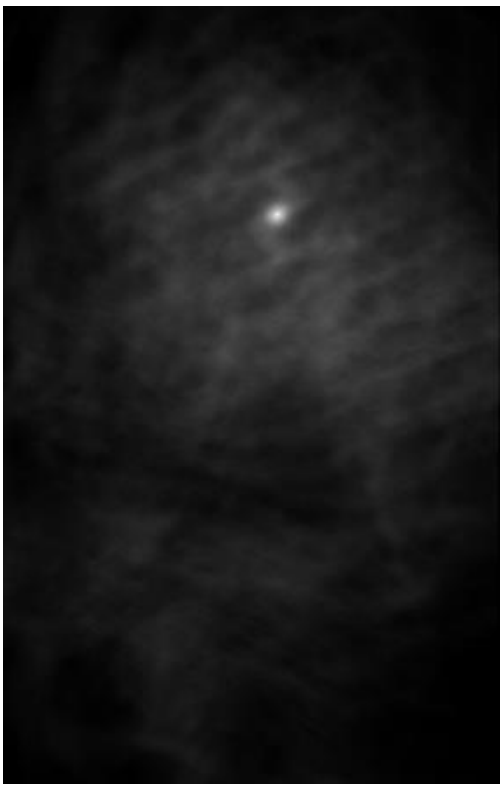
Obrázek 8.9 Znázornění přilehlého okolí pro velikost posunu vzoru po obrazu o 3px a 5px



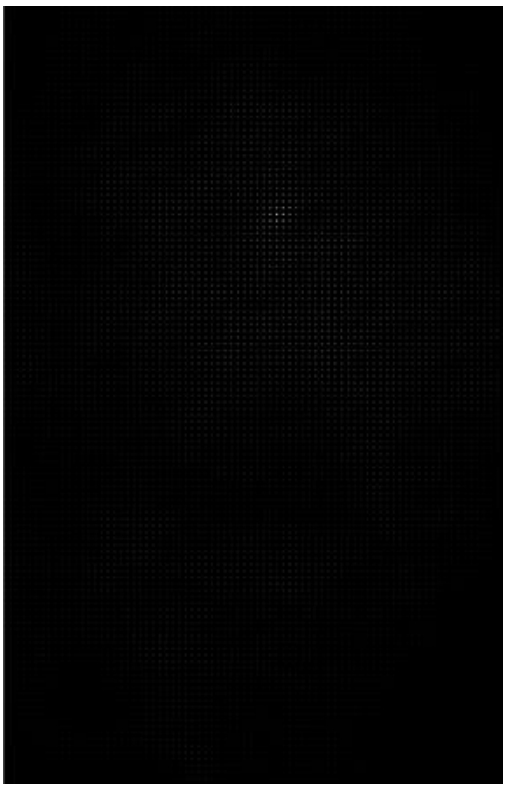
Obrázek 8.10 Obraz, na kterém bylo prováděno měření



Obrázek 8.11 Vzor, pomocí kterého bylo prováděno měření



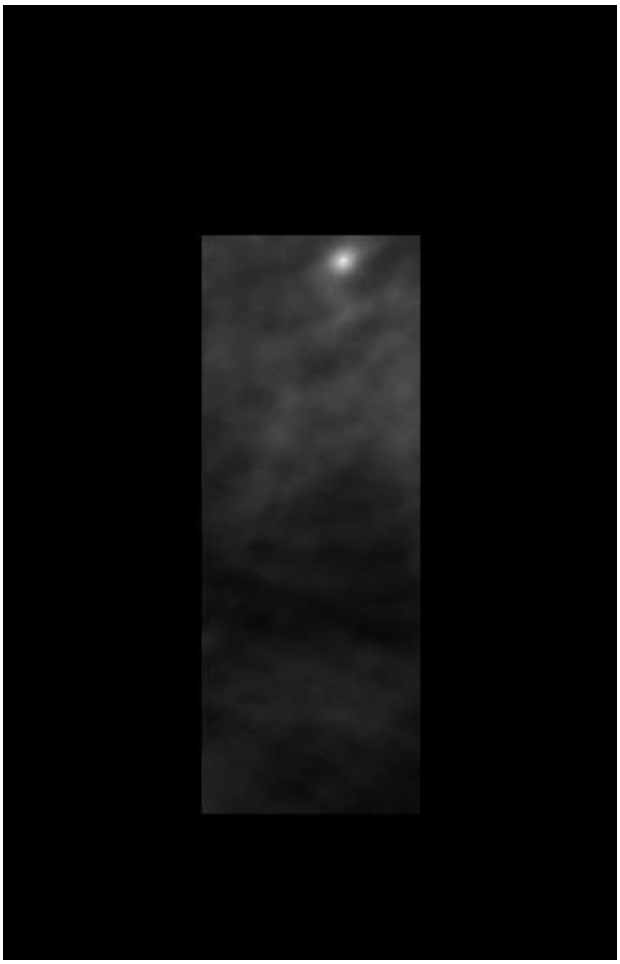
Obrázek 8.12 Akumulátor při kroku po 1px



Obrázek 8.13 Akumulátor při kroku po 4px

Při hledání se předpokládá, že hledaný vzor je celý. Dochází tak k tomu, že jsou v obrazu prohledávány oblasti, ve kterých se vzor nemůže nacházet. Jedná se především o rohy obrazu a jeho strany. V těchto místech se může nacházet jen část vzoru a je zbytečné tyto prostory prohledávat. Pokud je například vzor umístěn do rohu obrazu, tak se v něm nachází pouze ¼ vzoru. Nepočítání s těmito oblastmi výrazně zrychlí prohledávání.

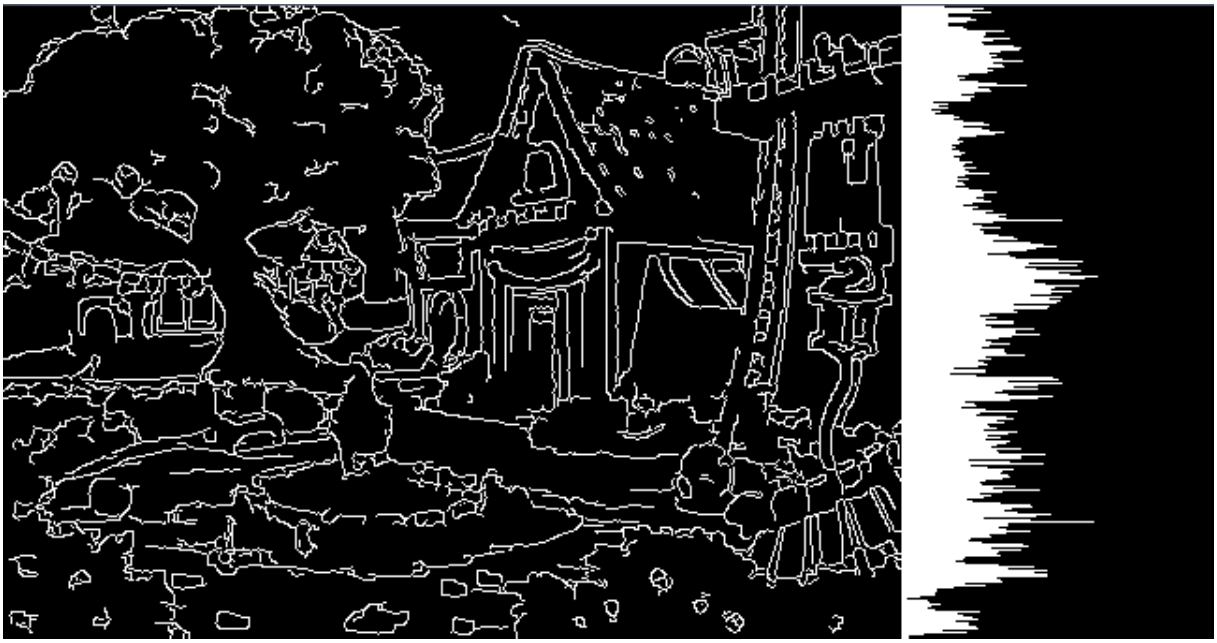
Při přípravě vzoru je pro něj definován střed, od kterého je poté možné získat vzdálenosti do všech čtyř směrů. Těmito vzdálenostmi je poté omezena prohledávaná plocha obrazu, viz obrázek 8.14. Při změně natočení vzoru se ale tyto vzdálenosti změní, proto je nutné je přepočítat buď pomocí goniometrických funkcí, anebo lze využít knihovnu OpenCV.



Obrázek 8.14 Akumulátor bez prohledaných krajních oblastí

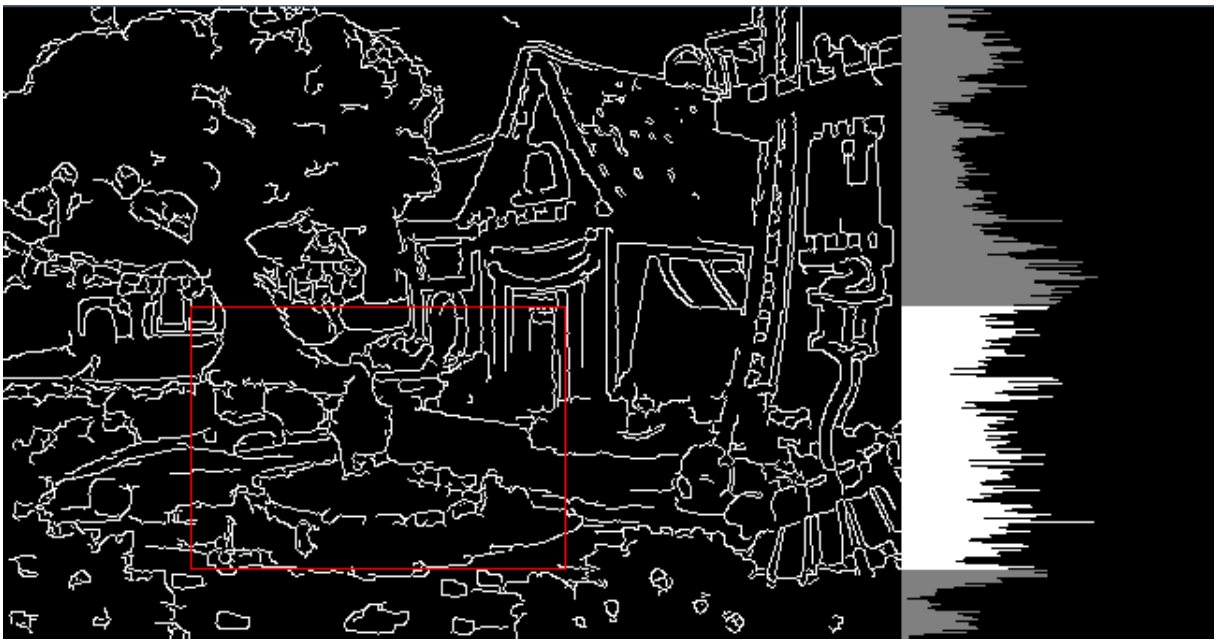
Pro další zrychlení vyhledávání hraje hlavní roli předpříprava obrazu. Při prohledávání obrazu se musí přejít přes všechny jeho body, jak černé tak i bílé. Při porovnávání se ovšem počítá pouze s bílými body a zbytečné procházení černých bodů způsobuje další zpomalení. Jak bylo zmíněno v kapitole 7.1, při předpřípravě dochází k uložení pozic všech bílých bodů, se kterými se poté počítá, a nedochází tak k procházení i černých bodů.

Důležité je také, jakým způsobem se tyto bílé body uloží. Ve většině případů bude hledaný vzor pouze z části obrazu, nikoli z celého, a tak pokud je vzor umístěn do levého horního rohu, je zbytečné pracovat s bílými body obrazu z pravého dolního rohu. Je tedy vhodné, aby se počítalo pouze s body v obraze, které leží v prostoru vzoru. Pokud by byl seznam bílých bodů uložen jen do jednorozměrného pole, nebylo by možné efektivně vyselektovat body, které se nachází v prostoru vzoru. Proto jsou bílé body z obrazu uloženy do dvourozměrného pole, jehož struktura je znázorněna na obrázku 8.15. Každý řádek z obrazu má vlastní pole, ve kterém se nachází bílé body z daného řádku.



Obrázek 8.15 Grafická reprezentace dvourozměrného pole pro bílé body z obrázku vlevo

Z této struktury je tak možné vybrat pomocí indexů pole, které odpovídají řádkům v obraze, pro dané umístění vzoru, viz obrázek 8.16. Posledním krokem je vyselektování bodů z pole pro dané řádky. Z levé strany je vybrán první bod, který náleží do oblasti vzoru pomocí algoritmu půlení intervalů, který je také znám jako Binární vyhledávání. Od tohoto bodu v obrazu dochází postupně k porovnávání bodů se vzorem. Jakmile se bod z obrazu nachází mimo prostor vzoru, dojde k ukončení porovnávání pro daný řádek a pokračuje se tak vyselektováním bodů pro další řádek, viz obrázek 8.17.



Obrázek 8.16 Vybrání polí pro řádky dle umístění vzoru

Obrázek 8.17 Vybrání bodů z vybraných polí pro řádky dle umístění vzoru

Při dané změně velikosti vzoru o 15px při vyhledávání dochází k určité nepřesnosti. Tato nepřesnost může způsobit, že správný obraz pro daný vzor může mít shodu jen například 80 %. To může vyústit v nepřesnost při hledání správného obrazu, a tak i chybné identifikaci. To je dáno především tím, že obrazy na kterých byl aplikován hranový detektor, mohou mít různý počet bílých bodů, od tisíců do deseti tisíců. Chyby nastávají především v případech, kdy vzor patří obrazu s malým množstvím bílých bodů. Když dochází k porovnávání

s obrazem, s velkým počtem bodů, vybraná oblast pro porovnávání má obvykle mnohem více bodů než vzor. Právě tato velká hustota bodů v dané oblasti může mít větší shodu než oblast ve správném obrazu a způsobuje tak špatnou identifikaci.

Je tedy vhodné prohledávat pouze oblasti v obrazech, které mají stejné, anebo podobné množství bílých bodů. Před porovnáváním je tedy nejprve spočteno množství bílých bodů v obraze pro oblast, která odpovídá vzoru. Pokud je maximální počet bílých bodů v dané oblasti menší než násobek 1,8 množství bodů vzoru, pak je prováděno samotné porovnávání pomocí obecné Houghovy transformace. V opačném případě se s danou oblastí nepočítá a vzor je posunut na další místo v obraze. Hodnota 1,8 byla stanovena na základě testů. Toto je prováděno pro všechny pozice vzoru v obraze. Důležitá je ale také efektivita spočtení bílých bodů. Jak bylo zmíněno v kapitole 7.1, z každého obrazu, na který byl aplikován hranový detektor, je spočten integrální obraz, viz kapitola 5.1. Pokud je černá barva v obraze reprezentována hodnotou 0 a bílá barva hodnotou 1, pak lze pomocí integrálního obrazu zjistit počet bílých bodů v dané oblasti. Pro spočtení bílých bodů je tedy použit integrální obraz. Tato optimalizace má za následek zpřesnění vyhledávání, a protože jsou omezeny výpočty pro nevhodné oblasti, dochází také i ke zrychlení.

8.1.2 SURF

Při zvolení SURF algoritmu pro vyhledávání je prvním krokem převedení vzoru do odstínu šedi, protože SURF algoritmus pracuje pouze s těmito obrazy. Dalším krokem je detekce významných bodů ve vzoru. To je provedeno podle postupu, který je uveden v kapitole 5.2. Posledním krokem je výpočet deskriptorů pro všechny nalezené významné body, pomocí kterých dochází k porovnávání obrazů se vzorem. To je provedeno podle kapitoly 5.4. Pro převod vzoru do odstínu šedi, nalezení významných bodů ve vzoru a vypočítání jejich deskriptorů je využita knihovna OpenCV.

Při ukládání obrazů do databáze dochází k předpočtení klíčových bodů a deskriptorů. Díky tomu je třeba už jen porovnat deskriptory obrazů s deskriptory vzoru. To je provedeno opět za pomoci OpenCV knihovny. Párování významných bodů podle deskriptorů u vzoru a obrazu je znázorněno na obrázkách 8.19 až 8.21. Výsledkem je pole, které vyjadřuje míru shody mezi deskriptory obrazu a vzoru. Ta je vyjádřena hodnotami od 0, která udává nejlepší shodu, do 1, která naopak vyjadřuje nejhorší shodu. Je tedy třeba toto pole přefiltrovat. To je provedeno tak, že je určen práh a pokud hodnoty spadají pod jeho hodnotu, pak jsou prohlášeny za správně identifikované. Hodnota pro tento práh byla zvolena pomocí testování na hodnotu 0,3. Posledním krokem je poté převedení na procenta počet hodnot z pole, které spadají pod práh 0,3 oproti celkovému množství hodnot z pole. A pomocí této procentuální hodnoty, která vyjadřuje míru shody, dochází k řazení obrazů.

Obrázek 8.19 Hledaný obraz

Obrázek 8.20 Vzor použitý pro hledání

Obrázek 8.21 Grafické zobrazení klíčových bodů ve vzoru a v obraze a jejich párování

8.2 Popis tříd

Serverová aplikace je složena z pěti balíčků. Balíček server.core obsahuje třídy pro správné fungování serveru a připravuje vzor pro vyhledávání. Balíček server.objectdetection obsahuje třídy, které implementují vyhledávací algoritmy SURF z knihovny OpenCV a vlastní implementaci obecné Houghovy transformace. Balíček server.admin obsahuje třídy pro komunikaci s administrátorskou aplikací a pro předpřípravu obrazových dat využitých pro vyhledávání. Další balíček server.gui obsahuje třídy grafického uživatelského rozhraní. Poslední balíček server.gui.img obsahuje sadu ikon, které jsou v aplikaci použity. V následujících kapitolách jsou popsány třídy, které dané balíčky obsahují. Class diagramy jsou součástí přílohy.

8.2.1 Balíček server.core

- ClientInfo – Pomocná třída pro uchování informací o připojeném klientovi jako je host name, host adresa a datum připojení klienta k serveru.

- Config – Třída pro uložení lokálních konfiguračních dat. Těmito daty je připojení k databázi, port, na kterém server běží, nastavení algoritmu pro vyhledávání a počtu vláken, které mají být využity pro vyhledávání. Součástí třídy jsou dvě metody pro uložení a načtení konfiguračního objektu.

- Console – Třída slouží pro vypisování řetězců do konzole aplikace pomocí dvojice statických metod print a println.

- Core – Hlavní třída, kdy při každém připojení klienta je pro tohoto klienta vytvořena nová instance. Té jsou pak předány obrazy načtené z databáze a hledaný vzor. Třída má dále na starosti přípravu vzoru pro jednotlivé vyhledávající algoritmy. Pro přípravu vzoru pro algoritmus obecné Houghovo transformace slouží metoda buildRTable a pro přípravu vzoru pro SURF algoritmus slouží metoda buildSurfPattern.

- DBConnection – Třída realizuje připojení k databázi. Dále třída vrací všechny metadata o obrazech uložené v databázi a data z konfigurační tabulky.

- PatternDetectionServer – Třída inicializuje OpenCV knihovnu a metodu main, která načítá konfigurační objekt a inicializuje grafické uživatelské rozhraní.

- Server – Třída, která čeká na připojení klienta a stará se o veškerou komunikaci mezi klientem a serverem. Řídí zpracování hledaného vzoru a samotné vyhledávání. Síťová komunikace probíhá pomocí třídy Socket.

- UtilsL – Pomocná třída, která obsahuje metodu pro vykreslení obrazu z objektu Mat a několik pomocných metod, které využívá obecná Houghova transformace a SURF algoritmus.

8.2.2 Balíček server.objectdetection

- Accumulator – Třída realizující akumulátor pro obecnou Houghovo transformaci. Každá instance akumulátoru popisuje právě jedno porovnání mezi vzorem a obrazem. Pro každé jiné natočení a změnu měřítka je třeba vytvořit další instanci akumulátoru. Informace o změně měřítka vzoru a natočení se také ukládají to této třídy. Tato třída je použita pouze pro testovací účely a pro vizualizaci výstupu. Kvůli velké paměťové náročnosti je ve finální verzi aplikace akumulátor odstraněn a nahrazen proměnnou, do které se ukládá největší hodnota, která byla pro akumulátor vypočtena.

- AccumulatorBestValue – Pomocná třída pro akumulátor, která uchovává největší hodnotu z daného akumulátoru.

- Ght – Implementace algoritmu pro obecnou Houghovo transformaci, jehož součástí je algoritmus pro třídění obrazů, které mají nejlepší shodu se vzorem. Tato třída je rozšířena abstraktní třídou ObejctDetection.

- GhtMapper – Pomocná třída pro obecnou Houghovo transformaci, která vypočítává všechny možnosti pro natočení a změnu velikosti pro vzor.

- AtanTable – Třída do které se uloží předpočtené hodnoty získané z funkce atan2, kterou využívá obecná Houghova transformace.

- Image – Instance této třídy nese informace právě o jednom obrazu, jako jsou id, název, popis, atd. Dále nese uložený obraz v poli bytů a předpočtené klíčové body a deskriptory pro SURF algoritmus. Pro ukládání akumulátoru pro obecnou Houghovo transformaci slouží list akumulátorů. V případě, že se akumulátory neukládají a ukládá se pouze jejich nejlepší hodnota (z důvodu úspory paměti), využívá se třída AccumulatorBestValue. Dále obsahuje vypočítaný integrální obraz z daného obrazu.

- ObjectDetection – Abstraktní třída, která rozšiřuje vyhledávací algoritmy. Má předdefinované hlavičky některých metod a implementované metody pro manipulaci s vlákny, jako je pozastavit, pokračovat anebo ukončit běh všech vláken.

- Surf – Implementace SURF algoritmu, jehož součástí je algoritmus pro třídění obrazů, které mají nejlepší shodu s obrazem. Je rozšířena abstraktní třídou ObjectDetection.

- SurfPattern – Pomocná třída pro SURF algoritmus, která uchovává informace o vzoru, jako jsou klíčové body a jejich deskriptory. Obsahuje také uložený vzor v barevné i černobíle podobě.

8.2.3 Balíček server.admin

- AdminServer – Třída rozšířena třídou Thread. Po připojení aplikace pro administraci je vytvořena nová instance této třídy, která se stará o komunikaci právě s jednou administrátorskou aplikací. Díky tomu, že běží ve svém vlastním vlákně, umožňuje připojení libovolného počtu administrátorských aplikací.

- ImageProcessor – Třída se stará o předpřípravu dat z obrazů, které jsou využity při vyhledávání. Těmito předpřipravenými daty jsou referenční tabulky, deskriptory a integrální obrazy.

8.2.4 Balíček server.gui

- AccumulatorWindow – Třída rozšířená třídou JFrame. Slouží pro zobrazení akumulátoru a používá se především pro testovací účely. Skládá se z listu, ve kterém se zobrazuje seznam všech akumulátorů pro daný obraz a vykreslovací plochy realizované pomocí třídy AccumulatorDrawPanel.

- AccumulatorDrawPanel – Jednoduchá třída rozšířená třídou JPanel, která slouží pro vykreslování akumulátoru.

- DetailWindow – Třída rozšířená třídou JFrame. Slouží především pro testovací účely. Zobrazuje seznam všech obrazů a zobrazovací plochy pro náhled obrazů. V případě že je dostupný akumulátor, tak po dvojkliku na název obrazu se zobrazí AccumulatorWindow.

- DrawPanel – Jednoduchá třída rozšířená třídou JPanel a slouží pro zobrazení vybraného obrazu ze třídy DetailWindow.

- MainWindow – Třída rozšířená třídou JFrame. Tato třída představuje hlavní okno, které je rozděleno pomocí třídy TabbedPane. V jedné záložce se zobrazuje konzole a ve druhé se zobrazuje seznam běžících úloh.

- SettingsWindowDB – Třída rozšířená třídou JFrame, která slouží pro nastavení připojení k databázi.

- SettingsWIndowOD - Třída rozšířená třídou JFrame, která slouží pro nastavení vyhledávacího algoritmu a počtu vláken, které mají vyhledávání provádět.

9 Klientská aplikace pro Android

Tato aplikace je určena pro zařízení s operačním systémem Android. Hlavní obrazovku lze logicky rozdělit do třech částí, viz obrázek 9.2. V horní části obrazovky se nachází dvojice tlačítek. První tlačítko slouží pro spuštění vyhledávání a druhé pro výběr hledaného vzoru. Ten lze vybrat z galerie mobilního zařízení, nebo je možné ho pořídit pomocí fotoaparátu. Část uprostřed je použita pro zobrazení nalezených obrazů a část v dolní části obrazovky zobrazuje informaci o tom, zda-li již byl vzor vybrán. Pokud je spuštěno vyhledávání, tato oblast se změní, viz obrázek 9.3. Zde je zobrazena informace o tom, že probíhá hledání a také je přidána dvojice tlačítek pro pozastavení, anebo úplné ukončení vyhledávání. Aplikace má ještě druhou obrazovku, která slouží pro nastavení připojení k serveru, viz obrázek 9.5.

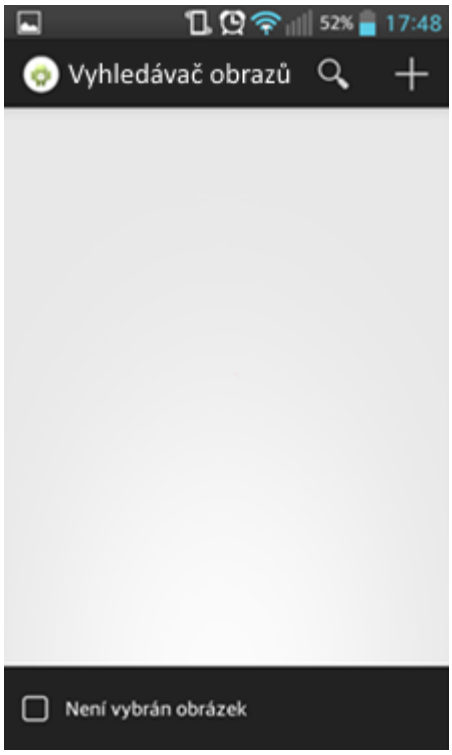
Při vybrání vzoru z galerie, nebo pořízením pomocí fotoaparátu je třeba, aby se daný vzor načetl do paměti. Mobilní zařízení ale obvykle nedisponují velkou pamětí a pořízené obrázky mají většinou velké rozlišení. Pokud by chtěl uživatel vyhledávat více vzorů po sobě, došlo by k zaplnění paměti telefonu a pádu aplikace. Serverová aplikace potřebuje vzor, který má největší stranu velikou maximálně 480px, protože uložené obrazy v databázi mají největší rozlišení právě 480px. Pro reálné použití může být vzor ale i menší. Jak bylo zmíněno v kapitole 8.1, velikost vzoru je stanovena na 300px. Tato velikost je dána kvůli ještě malému zkreslení a menšímu množství dat, které musí být přeneseny.

Pro načtení vzoru s menší velikostí do pamětí aplikace je použita třída Options z balíčku android.graphics.BitmapFactory. Nastavením parametru inJustDecodeBounds na hodnotu true dojde k tomu, že jsou načteny pouze rozměry vzoru. Po jejich načtení je nastaveno převzorkování obrazu pomocí inSampleSize. Poté je zavolána statická metoda decodeStream s příslušnými parametry a dojde k načtení vzoru s menším rozlišením, dle nastavené hodnoty inSampleSize.

```
BitmapFactory.Options bitmapOptions = new BitmapFactory.Options();
bitmapOptions.inJustDecodeBounds = true;
BitmapFactory.decodeStream(getContentResolver().openInputStream(uri), null, bitmapOptions);
bitmapOptions.inSampleSize = this.calculateInSampleSize(bitmapOptions, 300);
bitmapOptions.inJustDecodeBounds = false;
bitmapOptions.inPreferredConfig = Config.ARGB_8888;
Bitmap bitmap = BitmapFactory.decodeStream(getContentResolver().openInputStream(uri), null,
        bitmapOptions);
```

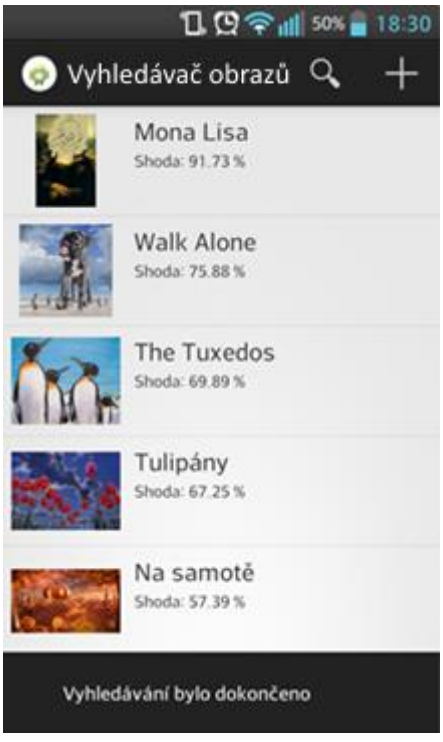
Obrázek 9.1 Kód pro změnu velikosti vzoru při načtení.

Po spuštění vyhledávání je tak zmenšený vzor odeslán na server, kde je spuštěno samotné vyhledávání. Postupně jsou na klienta odesílány mezivýsledky ze serveru, viz obrázek 9.4. Aplikace slouží především pro výběr vzoru a zobrazování výsledků. Neobsahuje tak žádnou složitější funkcionalitu.



Obrázek 9.2 Hlavní obrazovka

Obrázek 9.3 Hlavní obrazovka při spuštěném hledání



Obrázek 9.4 Hlavní obrazovka po skončení hledání

Obrázek 9.5 Obrazovka pro nastavení

9.1 Popis tříd

Klientská aplikace se skládá ze dvou balíčků classes, který obsahuje nejrůznější třídy, a balíčku id.activity, jehož jednotlivé třídy patří příslušným obrazovkám nazývající se Activity. Samotný design Aktivit se nachází ve zvláštní složce layouts. Class diagramy jsou součástí přílohy.

9.1.1 Balíček classes

- Client – Hlavní třída aplikace, která implementuje třídu Runnable. Po spuštění vyhledávání se vytvoří instance třídy Client, která se předá jako parametr novému vláknu. Samotná komunikace pak probíhá přes třídu Socket.

- ListAdapter – Pomocná třída která rozšiřuje BaseAdapter. Tato třída spravuje data pro ListView, který reprezentuje seznam nalezených objektů.

- Ping – Třída se stará o ověření připojení k serveru. To se provede pomocí třídy Socket a následně se spojení uzavře. Aby došlo k uzavření spojení i ze strany serveru, který čeká na data, pošle se enum hodnota PING. Server po zpracování této hodnoty ukončí spojení.

9.1.2 Balíček id.activity

- MainActivity – Hlavní obrazovka, ve které se vybírá hledaný vzor, a to buď z galerie, anebo prostřednictvím fotoaparátu. V dolní části aplikace se poté nachází tlačítka pro ovládání průběhu hledání.

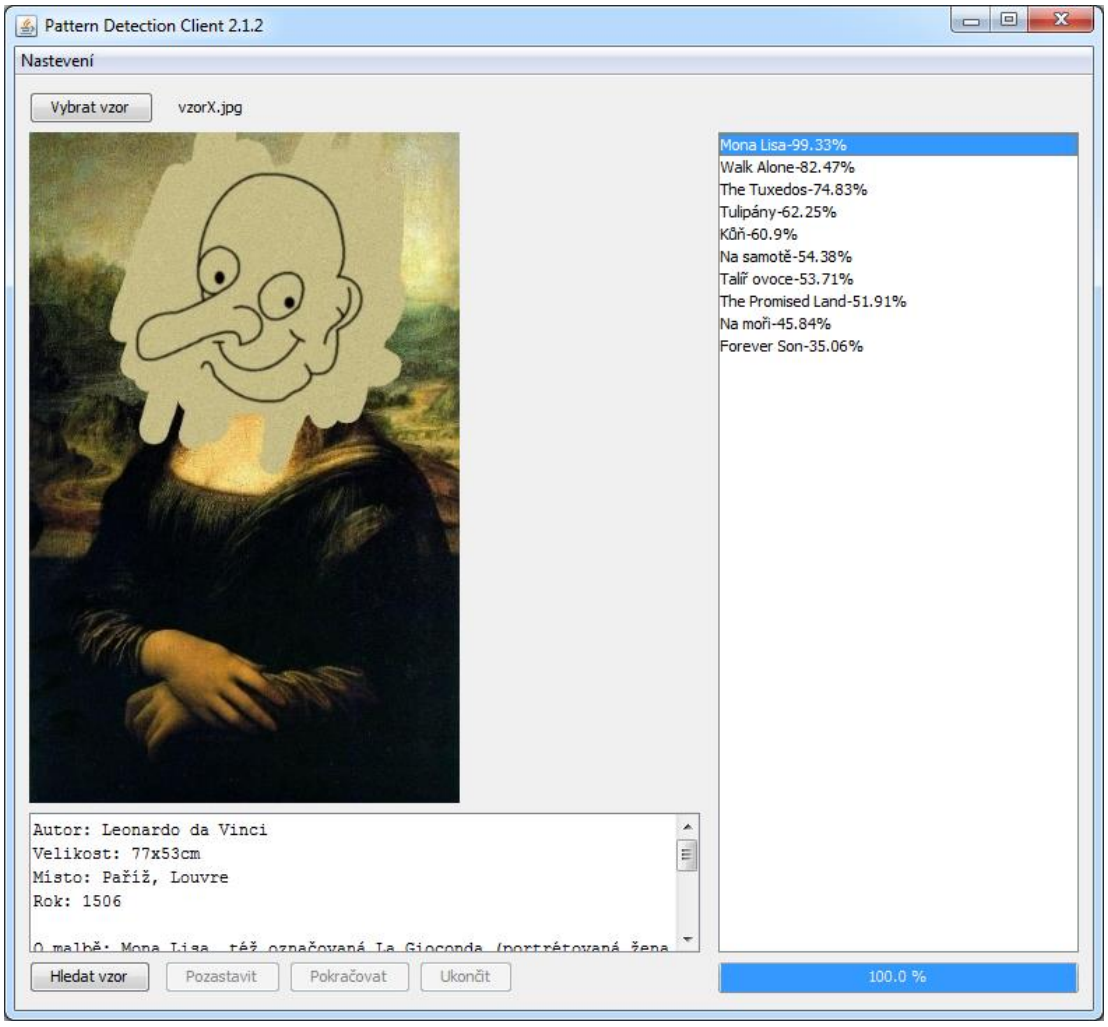
- DetailActivity – Obrazovka, která poskytuje náhled na vybraný obraz ze seznamu vykreslený v MainActivity. Kromě samotného obrazu obsahuje obrazovka další metadata k obrazu.

- SettingsActivity – Obrazovka sloužící pro nastavení připojení k serveru.

10 Klientská aplikace pro počítače

Tato aplikace slouží k rozpoznávání na počítačích. Její grafické rozhraní lze rozdělit do několika logických částí. V pravé části se zobrazuje seznam nalezených obrazů. Po vybrání jedné položky ze seznamu se zobrazí obraz v levé části, a pod ním jeho popis. Pro výběr vzoru slouží tlačítko „Vybrat vzor“. V horní části obrazovky se nachází jednoduché menu, které má pouze jednu položku, pomocí které se nastavuje připojení k serveru. V dolní části obrazovky se nachází čtveřice tlačítek pro ovládání vyhledávání a v pravém dolním rohu progress bar, který ukazuje v procentech, kolik zbývá do dokončení vyhledávání.

Funkcionalita aplikace je stejná jako u aplikace pro zařízení s operačním systémem Android. Po výběru vzoru dojde ke změně jeho velikosti na 300px. Důvod pro tuto velikost je popsán v předchozí kapitole. Změna velikosti není tak komplikovaná jako u aplikace pro operační systém Android a je použita pouze metoda resize z knihovny OpenCV. Po spuštění vyhledávání je vzor odeslán na server, kde je spuštěno samotné vyhledávání. Aplikace již pouze čeká na příchozí zpracované mezivýsledky, které zobrazuje v seznamu v pravé části obrazovky.



Obrázek 10.1 Hlavní obrazovka

10.1 Popis tříd

Klientská aplikace se skládá ze dvou balíčků client.core, který obsahuje základní logiku aplikace a balíček client.gui, který obsahuje grafické uživatelské rozhraní. Class diagramy jsou součástí přílohy.

10.1.1 Balíček client.core

- Client – Hlavní třída aplikace, která se stará o připojení k serveru pomocí třídy Socket. Dále přijímá průběžné mezivýsledky odesílané serverem a stará se o jejich zobrazení.
- Config – Třída pro uložení lokálních konfiguračních dat. Těmito daty je host a port serveru, na který se klient připojuje. Součástí třídy jsou dvě metody pro uložení a načtení konfiguračního objektu.
- Core – Třída mění velikost vzoru do předem definovaných hodnot a tento vzor je zde uchován pro další použití.
- Image – Třída, jejíž instance reprezentuje právě jeden obraz. Obsahuje samotný obraz, metadata a informace o nalezené shodě se vzorem, jako jsou souřadnice, natočení a změna měřítka, jsou-li k dispozici.
- PatternDetectionClient – Třída inicializuje OpenCV knihovnu a metodu main, která načítá konfigurační objekt a inicializuje grafické uživatelské rozhraní.
- UtilsL – Pomocná třída obsahující statickou metodu pro vykreslování obrazů z proměnné Mat používanou OpenCV knihovnou.

10.1.2 Balíček client.gui

- MainWindow – Třída rozšířená třídou JFrame. Slouží pro zobrazení obrazů, které mají nejlepší shodu se vzorem. List obsahující seznam nalezených obrazů se nachází na pravé straně. Na levé straně se zobrazuje náhled na vybraný obraz z listu. Na spodní straně se nachází pole pro zobrazení metadat o obraze a několik tlačítek pro ovládání vyhledávání. V horní části se nachází jednoduché menu s nastavením pro připojení k serveru.
- MainWindowDrawPanel – Třída rozšířená třídou JFrame, která slouží pro vykreslování obrazu z objektu Mat ve třídě MainWindow.
- SettingWindowServer – Třída rozšířená třídou JFrame, která slouží k nastavení připojení k serveru. Obsahuje dvě pole pro IP adresu a port serveru.

11 Knihovna

Tato vytvořená knihovna je jedním z důležitých prvků aplikace. Obsahuje několik tříd, které jsou společné pro všechny aplikace. Některé instance tříd pomáhají při síťové komunikaci a jiné pomáhají při výpočtech, které jsou stejné, nebo podobné v různých aplikacích. Knihovna se skládá z jednoho balíčku IDLibrary. Class diagram je součástí přílohy.

- Command – Třída implementuje rozhraní Serializable a obsahuje pouze výčtový typ enum, jehož instance se používají pro komunikaci mezi klientem a serverem.
- DataImageObject – Třída implementuje rozhraní Serializabel a slouží jako přepravní kontejner pro jeden obraz. Obsahuje pole bytů reprezentující daný obraz, rozměry obrazu, které se využívají při rekonstrukci obrazu z pole bytů, metadata obrazu, informaci o pozici, velikosti a natočení nalezeného vzoru v obrazu, jsou-li k dispozici, shodu se vzorem vyjádřenou v procentech a proměnnou datového typu HashMap pro případná další data.
- DataImageList – Třída implementuje rozhraní Serializable a slouží jako kontejner pro objekty datového typu DataImageObject. Všechny objekty realizující obrazy se pošlou najednou. Dále obsahuje proměnnou vyjadřující postup ve vyhledávání v procentech.
- DataPatterObject – Třída implementuje rozhraní Serializable a je použita pro přepravu vzoru od klienta na server. Obsahuje pole bytů představující vzor, jeho rozměry a informace o klientovi, který chce daný vzor vyhledávat jako je IP adresa a host name.
- ImageDataStore – Třída implementuje rozhraní Serializable a slouží pro uchovávání předpočtených dat obrazů, které vytváří administrace při jejich vkládání. Obsahuje informace o obrazech jako je jejich orientace, rozměry, seznam bílých bodů reprezentující hrany pro obecnou Houghovo transformaci, klíčové body a deskriptory pro SURF algoritmus.
- RTable – Třída implementuje rozhraní Serializable a slouží jako pomocná třída pro obecnou Houghovo transformaci. Třída uchovává informace o r-table.
- Utils – Třída obsahující pomocné statické metody, které se používají při změně velikosti obrazu, generování unikátního názvu obrazu, kontrolování zda je daný soubor obraz atd.

12 Testování

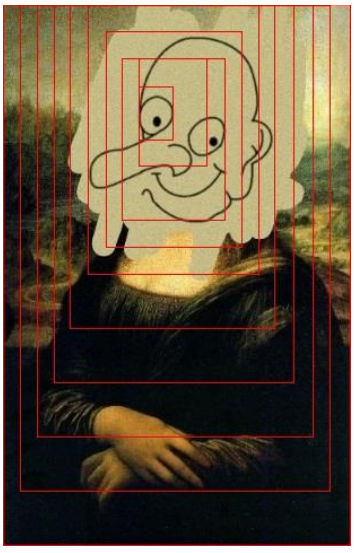
Pro testování funkčnosti aplikace byla vytvořena databáze obsahující 100 obrazů, ve kterých se bude vyhledávat. Obrazy a vzory použité pro testování jsou součástí přiloženého CD. Obrazy se nachází v adresáři „test/obrazy“ a vzory se nachází v adresáři „test/vzory“. Testována je především funkčnost použitých algoritmů pro vyhledávání. Těmi jsou obecná Houghova transformace a SURF algoritmus.

Testy jsou prováděny na počítači s operačním systémem Windows 7, procesorem i5 o frekvenci 1,6 GHz až 2,6 GHz s 8GB operační paměti DDR3.

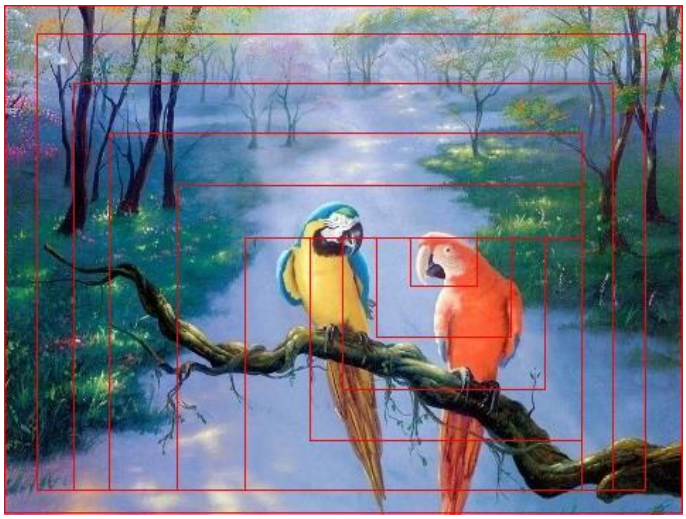
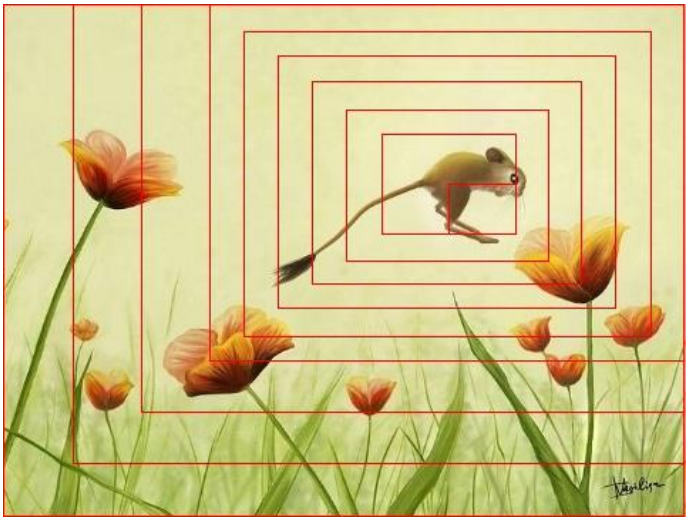
12.1 Testování závislosti velikosti vzoru na přesnosti vyhledávání

Pro testování bylo náhodně vybráno 10 obrazů z databáze. Z těchto obrazů byly pořizovány vzory, jejichž velikost se postupně zmenšovala o 10 %. Výsledky z testů jsou vidět v tabulce 12.1. V prvním sloupci se nachází obraz, ve kterém jsou červenou barvou vyznačeny polohy a velikosti vzorů, které byly v obraze vyhledávány. Ve druhém sloupci se nachází velikosti vzorů vyjádřené v procentech. Stejné velikosti obrazu a vzoru odpovídá hodnota 100 % a nejmenší velikost vzoru odpovídá hodnotě 10 %. Ve třetím a čtvrtém sloupci se nachází výsledky vyhledávání pro SURF algoritmus a obecnou Houghovu transformaci pro jednotlivé velikosti vzorů, které odpovídají druhému sloupci. Číslo před lomítkem udává, na jakém místě byl obraz nalezen, a číslo za lomítkem udává, v kolika obrazech byl vzor vyhledáván.

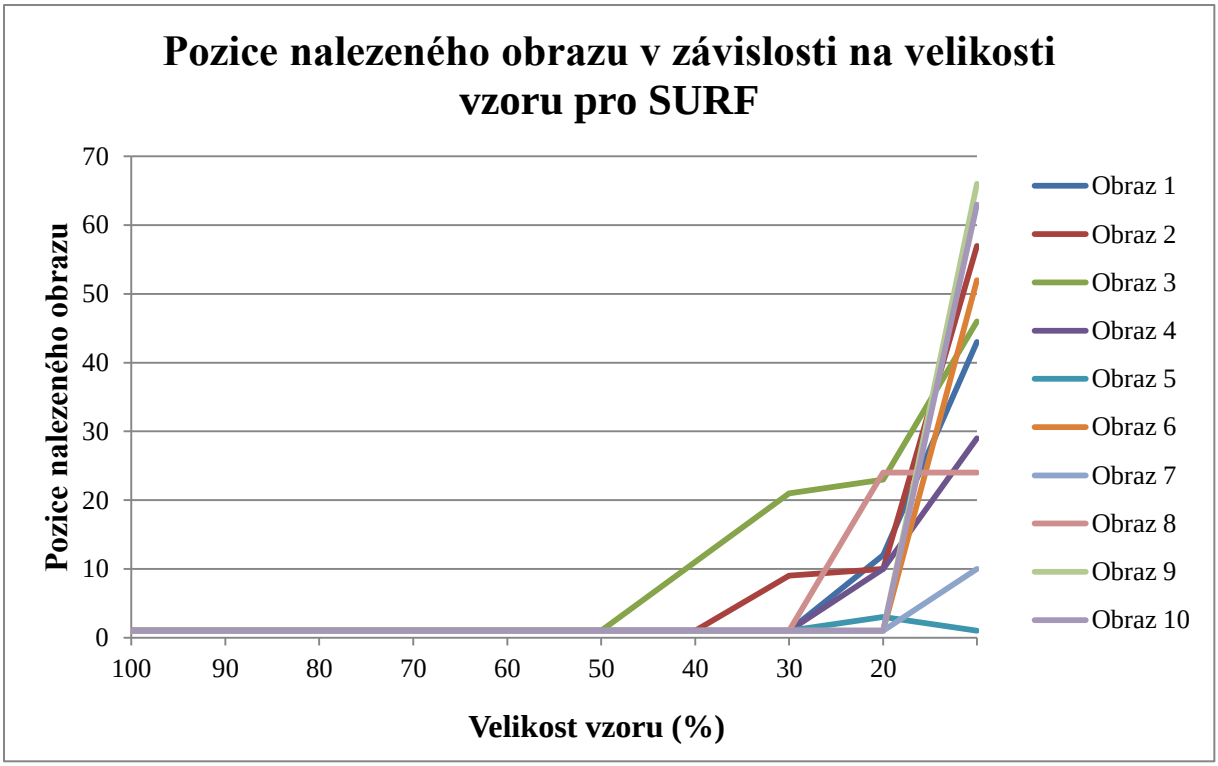
Obraz	Velikost vzoru	SURF	GHT
	100 %	1/100	1/100
	90 %	1/100	1/100
	80 %	1/100	1/100
	70 %	1/100	1/100
	60 %	1/100	1/100
	50 %	1/100	1/100
	40 %	1/100	1/100
	30 %	1/100	1/100
	20 %	12/100	1/100
	10 %	43/100	7/100

	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 % 10 %	1/100 1/100 1/100 1/100 1/100 1/100 1/100 9/100 10/100 57/100	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 2/100 57/100
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 % 10 %	1/100 1/100 1/100 1/100 1/100 1/100 11/100 21/100 23/100 46/100	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 5/100
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 % 10 %	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 10/100 29/100	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 27/100 51/100

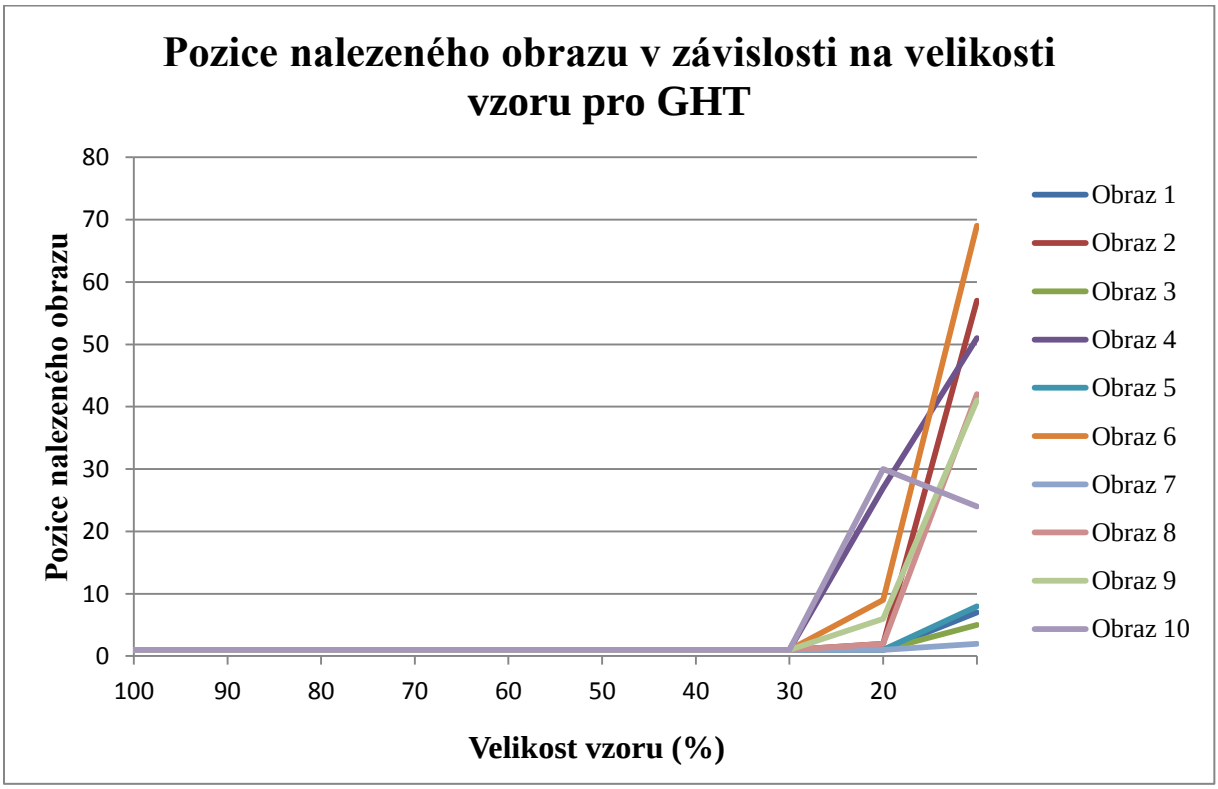
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 % 10 %	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 3/100 1/100	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 8/100
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 % 10 %	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 52/100	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 9/100 69/100
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 % 10 %	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 10/100	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 2/100

	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 % 10 %	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 24/100 24/100	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 2/100 42/100
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 % 10 %	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 66/100	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 6/100 41/100
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 % 10 %	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 63/100	1/100 1/100 1/100 1/100 1/100 1/100 1/100 1/100 30/100 24/100

Tabulka 12.1 Výsledky testů vyhledávání vzorů v obrazech



Obrázek 12.1 Graf znázorňující výsledky testování SURF algoritmu



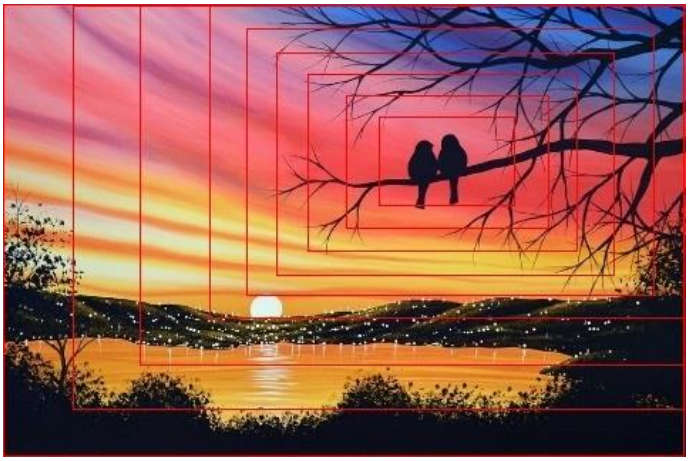
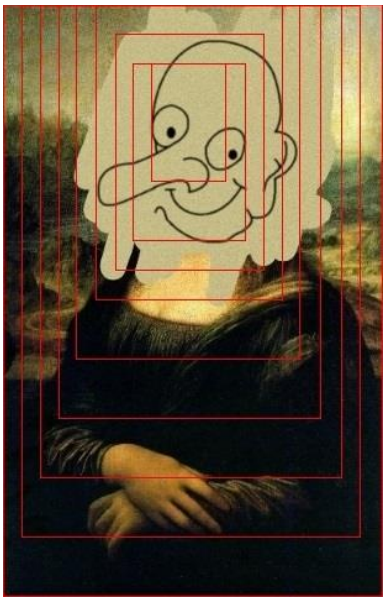
Obrázek 12.2 Graf znázorňující výsledky testování obecné Houghovy transformace

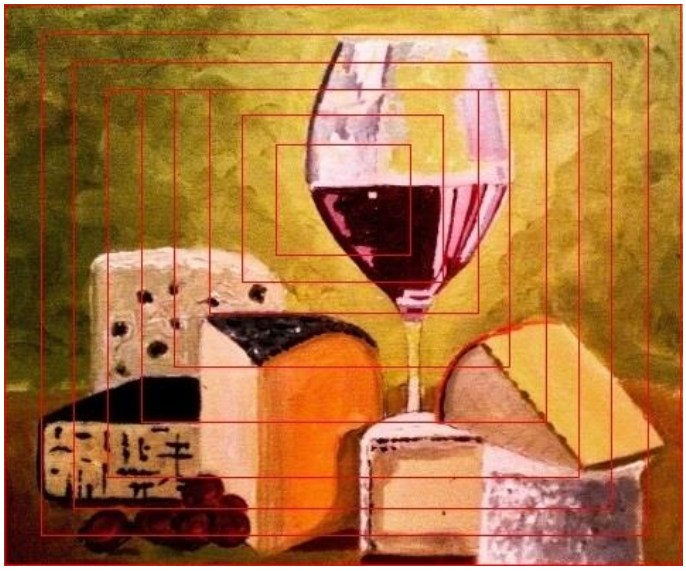
Z výsledků je zřejmé, že obecné Houghovy transformaci a SURF algoritmu stačí pro úspěšné identifikování obrazu vzor, jehož minimální velikost je přibližně 30 % velikosti obrazu. Důležitou roli zde hraje samotný obsah obrazu, díky čemuž může být vhodný pro vyhledávání jeden či druhý algoritmus. Proto minimální velikosti vzoru 30 % oproti obrazu, může ještě kolísat přibližně o 10 %. Aby se s jistotou dalo říci, o jaký obraz se jedná, je třeba vzor o minimální velikosti, která je přibližně 40 % obrazu.

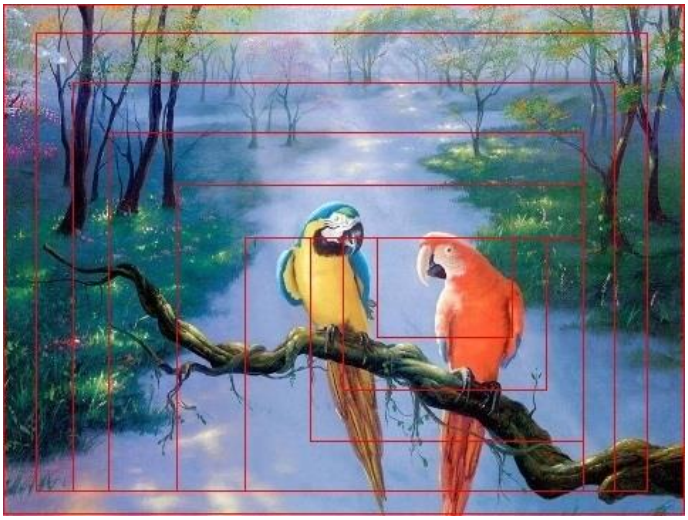
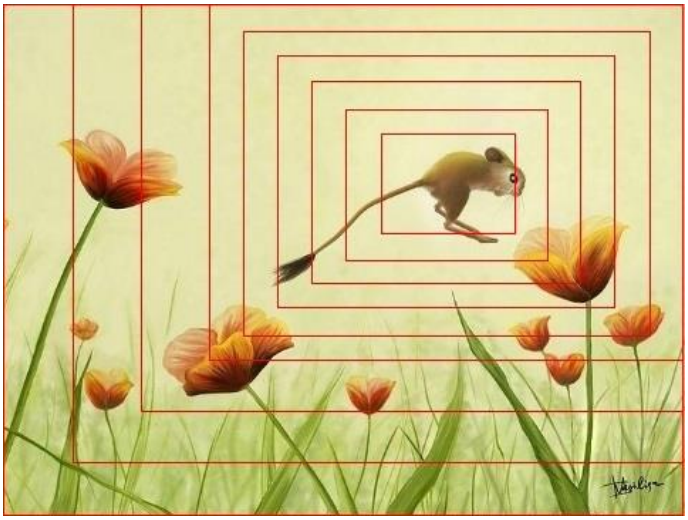
12.2 Testování závislosti velikosti vzoru na rychlosti vyhledávání

Pro testování bylo z databáze vybráno stejných 10 obrazů, jako v kapitole 12.1. Z těchto obrazů byly pořizovány vzory, jejichž velikost se postupně zmenšovala o 10 %. Výsledky z testů jsou vidět v tabulce 12.2. V prvním sloupci se nachází obraz, ve kterém jsou červenou barvou vyznačeny polohy a velikosti vzorů, které byly v obraze vyhledávány. Ve druhém sloupci se nachází velikosti vzorů vyjádřené v procentech. Stejné velikosti obrazu a vzoru odpovídá hodnota 100 % a nejmenší velikost vzoru odpovídá hodnotě 20 %. Z předchozí kapitoly je zřejmé, že velikost vzoru odpovídající 10 % již nemá dobré výsledky. Proto tato velikost vzoru není zahrnuta do dalšího testování. Ve třetím a čtvrtém sloupci se nachází časy, které reprezentuje dobu vyhledávání pro SURF algoritmus a obecnou Houghovu transformaci pro jednotlivé velikosti vzoru. Doba je vždy měřena od spuštění vyhledávání až po nalezení nejlepšího výsledku. V tabulce jsou časy zaznamenány v sekundách.

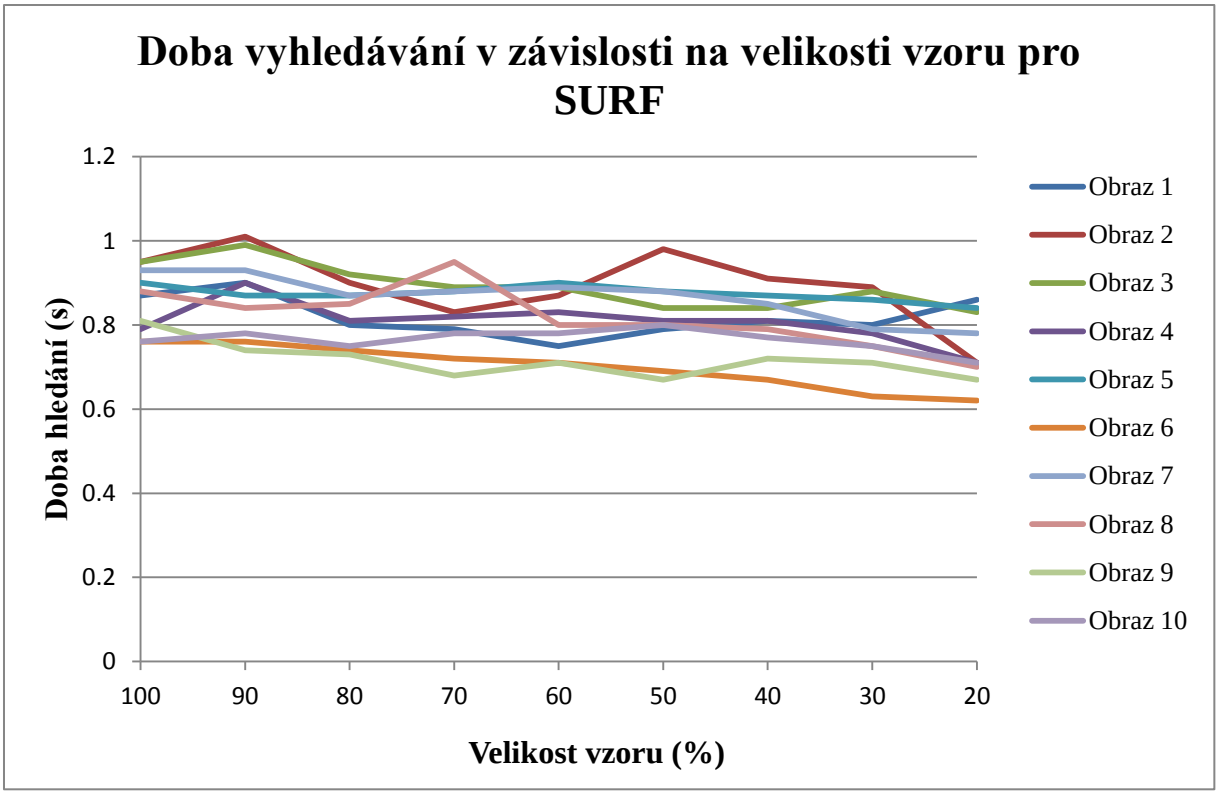
Obraz	Velikost vzoru	SURF	GHT
	100 %	0,87	2,1
	90 %	0,9	1,9
	80 %	0,8	1,4
	70 %	0,79	1,2
	60 %	0,75	1,5
	50 %	0,79	5
	40 %	0,81	6,5
	30 %	0,8	6,6
	20 %	0,86	10,5

	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 %	0,95 1,01 0,9 0,83 0,87 0,98 0,91 0,89 0,71	10,4 10,3 10,2 27,9 25,9 26,5 72,1 26,2 24,2
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 %	0,95 0,99 0,92 0,89 0,89 0,84 0,84 0,88 0,83	12,1 11,5 10,5 10,5 26,8 23,1 72,2 53,1 33,8
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 %	0,79 0,9 0,81 0,82 0,83 0,81 0,81 0,78 0,71	2,9 3,2 3,5 3,5 13,7 13,8 23,8 23,1 11,6

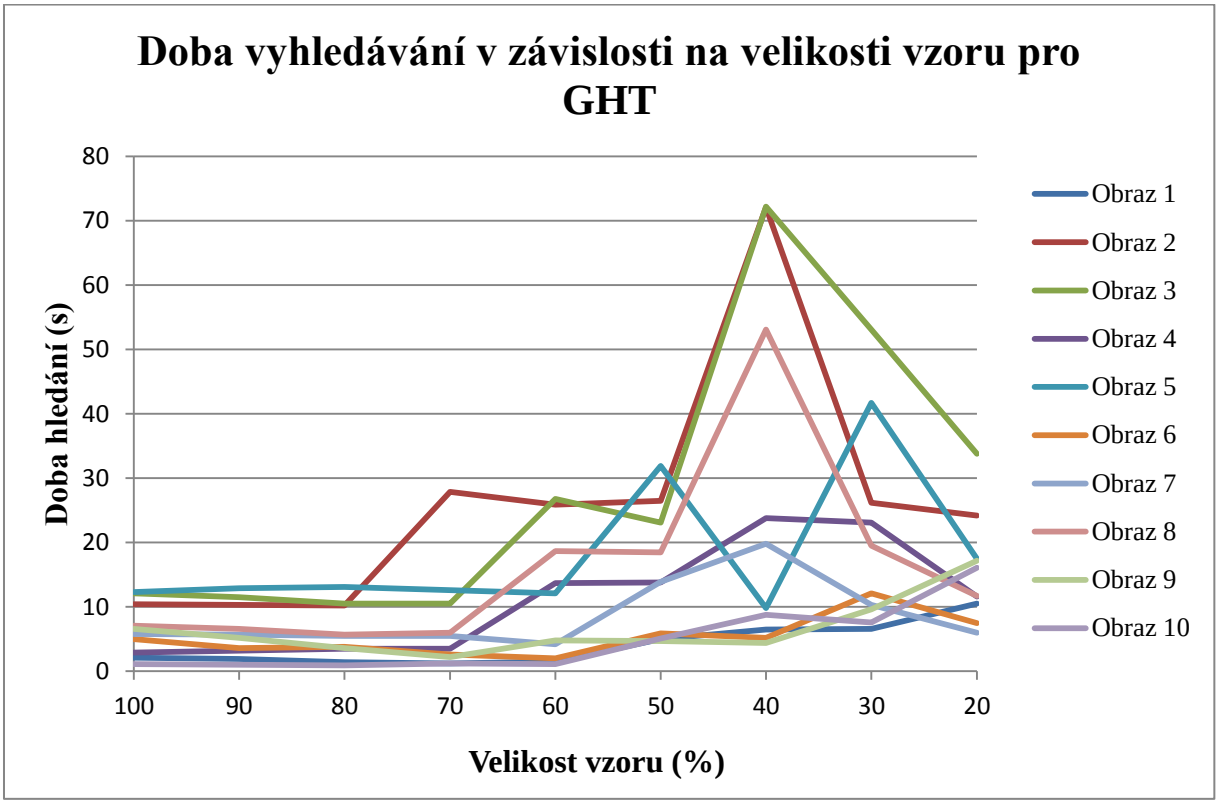
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 %	0,9 0,87 0,87 0,88 0,9 0,88 0,87 0,86 0,84	12,3 12,9 13,1 12,6 12,1 31,9 9,8 41,7 17,5
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 %	0,76 0,76 0,74 0,72 0,71 0,69 0,67 0,63 0,62	5 3,6 3,8 2,6 2 5,9 5,2 12,1 7,5
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 %	0,93 0,93 0,87 0,88 0,89 0,88 0,85 0,79 0,78	5,8 5,7 5,5 5,5 4,2 13,9 19,8 10,3 6

	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 %	0,88 0,84 0,85 0,95 0,8 0,8 0,79 0,75 0,7	7,1 6,6 5,7 6 18,7 18,5 53,1 19,5 11,7
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 %	0,81 0,74 0,73 0,68 0,71 0,67 0,72 0,71 0,67	6,6 5,2 3,6 2,2 4,8 4,7 4,4 9,6 17,2
	100 % 90 % 80 % 70 % 60 % 50 % 40 % 30 % 20 %	0,76 0,78 0,75 0,78 0,78 0,8 0,77 0,75 0,71	1,1 1 0,9 1,2 1,1 5,1 8,8 7,6 16,1

Tabulka 12.2 Výsledky testů vyhledávání vzorů v obrazech



Obrázek 12.3 Graf znázorňující výsledky testování SURF algoritmu



Obrázek 12.4 Graf znázorňující výsledky testování obecné Houghovy transformace

Při zjišťování času, který je potřebný pro nalezení obrazu podle vzoru, bylo vždy provedeno několik měření a průměrný čas byl zaznamenán do tabulky. U SURF algoritmu se měřené časy lišily od průměrné hodnoty o několik desetin sekundy a u obecné Houghovy transformace se měřené časy lišily od průměrné hodnoty i o několik sekund.

U SURF algoritmu se čas pro nalezení obrazu pohybuje vždy kolem 1 sekundy. Ve většině testovaných obrazů s klesající velikostí vzoru postupně klesá i doba potřebná pro vyhledávání. To je dáno tím, že se zmenšujícím se vzorem klesá také množství detekovaných klíčových bodů, které se používají pro porovnávání. Jak je možné vyčíst z tabulky 12.2 a grafu 12.3, pro dosažení největší rychlosti vyhledávání je třeba použít vzor o velikosti 20 % - 30 % velikosti obrazu.

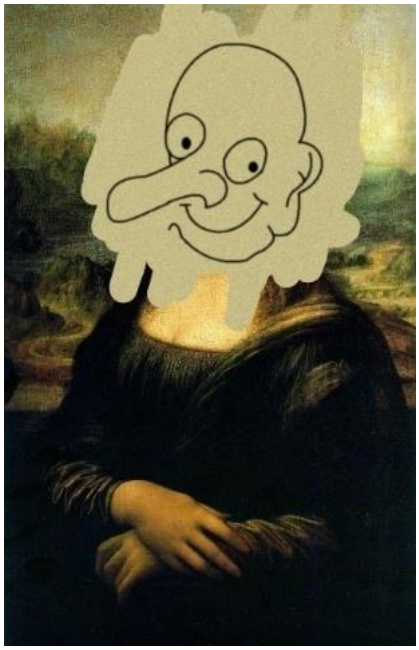
U obecné Houghovy transformace se čas pro nalezení obrazu mění od jednotek až po desítky sekund. To je dáno obsahem samotného obrazu. Pokud je obraz, ve kterém se hledá složitý, bude obsahovat i velké množství bílých bodů, které se používají pro porovnávání, a tím významně vzroste i doba hledání. Doba potřebná pro vyhledávání je oproti SURF algoritmu poměrně nestabilní, jak lze vyčíst z tabulky 12.2 a grafu 12.4. Při velké velikosti vzoru je třeba porovnat velké množství bodů. Vzor ovšem není třeba posouvat po celém obraze, viz kapitola 8.1.1. Se zmenšujícím se vzorem postupně klesá doba potřebná pro vyhledávání, ale také se postupně zvětšuje plocha, ve které se vzor vyhledává. Od určité velikosti vzoru začne zvětšující se plocha zpomalovat vyhledávání. Tato mezní velikost vzoru pro zpomalení vyhledávání je u každého obrazu jiná. V průměru se ale tato velikost vzoru pohybuje kolem 60 % velikosti obrazu, jak je možné vyčíst z tabulky 12.2 a grafu 12.4.

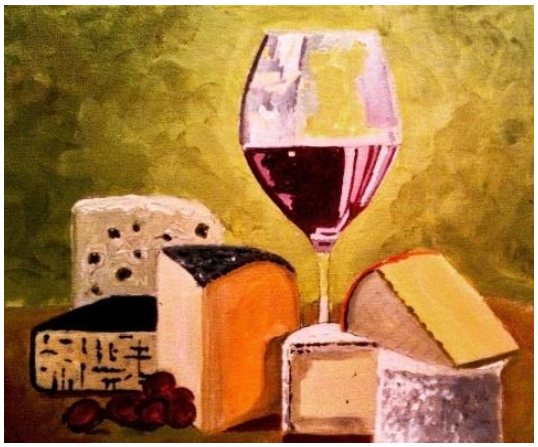
Z naměřených hodnot je zřejmé, že SURF algoritmus je mnohonásobně rychlejší než obecná Houghova transformace. Pro co nejrychlejší běh SURF algoritmu je potřeba použít vzor o přibližné velikosti 30 % velikosti obrazu a pro obecnou Houghovu transformaci je třeba vzor o přibližné velikosti 60 % velikosti obrazu.

12.3 Testování na reálných datech

Pro testování bylo z databáze vybráno stejných 10 obrazů, jako v kapitole 12.1 a 12.2. Z těchto obrazů, byly pořízeny vzory o různých velikostech pomocí fotoaparátu z mobilního telefonu s rozlišením 8MPx. Takto pořízené vzory byly poté vyhledávány pomocí daných algoritmů a výsledky zaneseny do tabulky 12.3. Tato tabulka se skládá ze čtyř sloupců. V prvním sloupci se nachází originální obraz. Ve druhém sloupci se nachází hledaný vzor. Ve třetím a čtvrtém sloupci se nachází výsledky vyhledávání pro SURF algoritmus a obecnou Houghovu transformaci. Číslo před lomítkem udává, na jakém místě byl obraz nalezen, a číslo za lomítkem udává, v kolika obrazech byl vzor vyhledáván.

Originální obraz	Hledaný vzor	SURF	GHT
		1/100	1/100
		1/100	1/100
		1/100	1/100

		1/100	1/100
		1/100	1/100
		1/100	1/100

		1/100	1/100
		1/100	1/100
		1/100	1/100

		1/100	1/100

Tabulka 12.3 Výsledky testů vyhledávání vzorů v obrazech

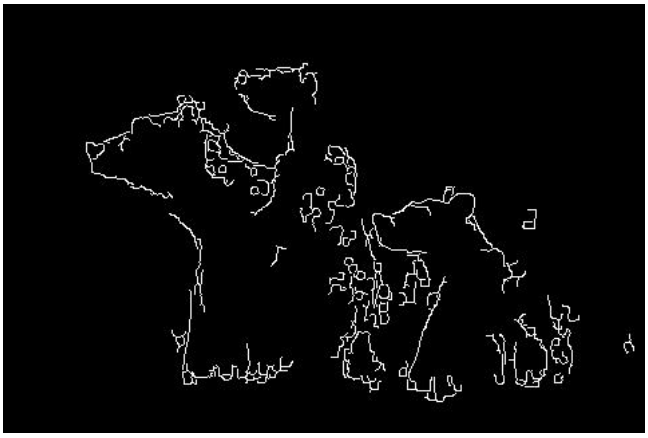
Z výsledků je zřejmé, že si oba algoritmy použité pro hledání vedly velice dobře. U SURF algoritmu i obecné Houghovy transformace je úspěšnost nalezení správného výsledku stoprocentní.

Z pozorování při provádění testů bylo dále zjištěno, že obecná Houghova transformace je velice náchylná na změnu osvětlení. Tato změna osvětlení může způsobit především to, že jsou po aplikování hranového detektoru ve vzoru detekovány jiné hrany, oproti originálnímu obrazu, viz obrázek 12.1 až 12.4. Tato změna má za následek velký pokles přesnosti.

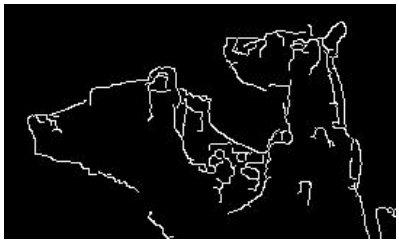
SURF algoritmus je oproti obecné Houghovy transformaci mnohem méně náchylný na změnu osvětlení a ve výsledné přesnosti se změna v osvětlení téměř neprojeví.

Obrázek 12.1 Originální obraz

Obrázek 12.2 Vzor se špatným osvětlením



Obrázek 12.3 Hrany u originálního obrazu



Obrázek 12.4 Hrany u vzoru se špatným osvětlením

13 Závěr

První část této práce byla zaměřena na teorii týkající se zpracování obrazu. Byly zde prozkoumány jednotlivé kroky, ze kterých se obvykle zpracování obrazu skládá. Těmito kroky jsou předzpracování obrazu, segmentace a rozpoznávání. U všech těchto kroků bylo vždy naznačeno několik metod, kterých by mohlo být využito. Blíže byly poté popsány právě ty metody, které jsou ve výsledné aplikaci použity. U předzpracování obrazu to byla eroze a dilatace. U segmentace bylo popsáno prahování, detekce hran a Cannyho hranový detektor. Pro rozpoznávání obrazu byly popsány dvě použité metody, obecná Houghova transformace a SURF algoritmus.

Druhá část práce byla zaměřena především na návrh aplikací, které budou pro rozpoznávání obrazu vytvořeny. Nejdůležitější je serverová aplikace, která má na starosti samotné vyhledávání obrazů. Pro správu obrazů pak slouží jednoduchá administrátorská aplikace. K dispozici jsou také dvě klientské aplikace, jedna pro klasické počítače a druhá pro mobilní zařízení s operačním systémem Android. Tyto aplikace jsou použity pro zobrazování nalezených výsledků z hledání.

Ve třetí části byly popsány funkcionality jednotlivých vytvořených aplikací, popsány všechny vytvořené třídy a znázorněny pomocí class diagramů. U části, zabývající se administrátorskou aplikací, byly především popsány způsoby předpřípravy obrazů. U další části, která se zabývá serverovou aplikací, byly podrobně popsány implementace použitých vyhledávacích algoritmů. U obecné Houghovy transformace, která byla vytvořena bez použití jakékoliv knihovny, byla popsána její implementace v aplikaci. Velká část byla také věnována optimalizaci tohoto algoritmu pro co nejrychlejší běh. Druhý použitý algoritmus SURF, je realizován pomocí knihovny OpenCV. Ten je již optimalizován, a proto zde byla popsána pouze jeho implementace v aplikaci. U dalších dvou částí, které se zabývají klientskými aplikacemi, byl vytvořen pouze jejich krátký popis, protože tyto aplikace jsou velice jednoduché a neobsahují žádnou složitou funkcionalitu.

Poslední část je zaměřena na testování použitých algoritmů pro vyhledávání. Při prvním testu byla zjišťována přesnost algoritmů v závislosti na velikosti vzoru. Oba testované algoritmy si vedly velice podobně. Z tohoto testu bylo zjištěno, že pro úspěšné nalezení obrazu je třeba vzor o přibližné minimální velikosti 30 %. Druhý test byl zaměřen na rychlost vyhledávání. Zde byl SURF algoritmus jednoznačně rychlejší, v krajních případech i o desítky sekund, oproti obecné Houghovy transformaci. Poslední test byl prováděn na reálných datech, kdy vzory byly pořizovány pomocí fotoaparátu. Takto pořízené vzory byly různě zkresleny. Za těchto podmínek si obecná Houghova transformace i SURF algoritmus vedli velice dobře. Oběma algoritmům se povedlo správně identifikovat všech 10 obrazů. Přesto bylo možné při testování vypozorovat, že je obecná Houghova transformace více náchylnější na změnu osvětlení, než SURF algoritmus.

Vytvořené aplikace je dále možné rozvíjet a optimalizovat. Lze se zaměřit na zmenšení citlivosti obecné Houghovy transformace na osvětlení. Také je možné zrychlit oba použité algoritmy pomocí GPU. Pro využití GPU značky NVIDIA lze využít knihovnu JCuda, nebo lze využít knihovny OpenCL pro GPU značky INTEL. Dále může být také rozšířena

administrátorská aplikace do mobilních zařízení s operačním systémem Android tak, aby uživatel mohl přidávat obrazy do databáze rovnou z těchto zařízení.

14 Použitá literatura

[1] I.T. Young, J.J. Gerbrands, L.J. van Vliet, Fundamental of Image Processing, Delf University of Technology, 2007 [Online]. Dostupné z: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/TUDELFT/FIP2_3.pdf

[2] J. Fiřt, R. Holota, Digitalizace a zpracování obrazu, Západočeská univerzita, Plzeň, 2002 [Online]. Dostupné z: http://home.zcu.cz/~holota5/publ/DigZprO.pdf

[3] R. C. Gonzales, R. E. Woods, Upper Saggle River, Digital image processing, 2008 [Online]. Dostupné z: http://lit.fe.uni-lj.si/showpdf.php?lang=slo&type=doc&doc=dip

[4] M. Železný, Zpracování digitalizovaného obrazu, Plzeň, 2001 [Online]. Dostupné z: http://www.kky.zcu.cz/uploads/courses/zdo/ZDO_aktual_130215.pdf

[5] S. Nimkar, S Shrivastava, S. Varghese, India, Maharashtra, Contrast enhancement and brightness preservation using multi-decomposition histogram equalization, 2013 [Online]. Dostupné z: http://arxiv.org/ftp/arxiv/papers/1307/1307.3054.pdf

[6] A. Procházka, Image segmentation for object detection, Praha, 2011 [Online]. Dostupné z: http://dsp.vscht.cz/konference_matlab/MATLAB11/prispevky/129_yadollahi.pdf

[7] MathWorks, Morphology Fundamentals: Dilation and Erosion [Online]. Dostupné z: http://www.mathworks.com/help/images/morphology-fundamentals-dilation-and-erosion.html

[8] O. Mikšík, gymnázium Kroměříž, Praktické využití metod digitálního zpracování obrazu, 2007 [Online]. Dostupné z: http://soc.nidv.cz/data/2007/01-2.pdf

[9] R. Fisher, S. Perkins, A.Walker, E. Wolfart, Image processing learning resources, England, the School of Informatics, 2003 [Online]. Dostupné z: http://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm

[10] T. Seemann, Australia, Digital Image Processing using Local Segmentation, Australia, 2002 [Online]. Dostupné z: http://www.csse.monash.edu.au/~torsten/pubs/Seemann-thesis.pdf

[11] ČVUT, Praha, 2011 [Online]. Dostupné z: http://www.fbmi.cvut.cz/files/predmety/3528/public/Metody%20rozpozn%C3%A1n%C3%AD%20objekt%C5%AF%20v%20obrazu.pdf

[12] J. Canny, A Computational Approach to Edge Detection, 1986 [Online]. Dostupné z: http://cmp.felk.cvut.cz/~cernyad2/TextCaptchaPdf/A%20Computational%20Approach%20to%20Edge%20Detection.pdf

[13] J. Pikora, Implementace grafických filtrů pro zpracování rastrového obrazu, Masarykova univerzita, Fakulta informatiky, Brno, 2008 [Online]. Dostupné z: http://is.muni.cz/th/60754/fi_b/bakalarka.pdf

[14] A. M. Khan, S. Ravi, Image Segmentation Methods, International Journal of Sorf Computing and Engineering, 2013 [Online]. Dostupné z: http://www.ijsce.org/attachments/File/v3i4/D1760093413.pdf

[15] R. Rao, Computer Vision, 2009 [Online]. Dostupné z: https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect6.pdf

[16] E. Turkel, Hough Transform [Online]. Dostupné z: http://www.math.tau.ac.il/~turkel/notes/HoughTransform.pdf

[17] P. Šťástný, Moderní metody identifikace objektů, Brno, 2011 [Online]. Dostuupné z: http://is.muni.cz/th/255824/fi_b_a2/Stastny.pdf

[18] Center for Machine Perception, Hledání parametrického popisu objektů pomocí Houghovy transformace, 2008 [Online]. Dostupné z: http://cmp.felk.cvut.cz/cmp/courses/ZS1/Cviceni/cv5/hough.htm

[19] T. Janovič, Detekce polohy očí v obrazech obličeje pomocí Houghovy transformace, Brno, 2012 [Online]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=54849

[20] D. H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, 1980 [Online]. Dostupné z: http://www.sci.utah.edu/~gerig/CS7960-S2010/handouts/Ballard-Generalized-HoughT.pdf

[21] H. Bay, A. Ess, T. Tuztelaars, L. Gool, Speeded-UP Robust Features (SURF) Switzerland, Zurich, 2010 [Online]. Dostupné z: https://www.vision.ee.ethz.ch/en/publications/papers/articles/eth_biwi_00517.pdf

[22] D. Lowe, Distinctive image features from scale-invariant keypoints, 2004 [Online]. Dostupné z: http://www.inf.fu-berlin.de/lehre/SS09/CV/uebungen/uebung09/SIFT.pdf

[23] Computer Vision – The Integral Image. Computer science source, 2010 [Online]. Dostupné z: https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/

[24] P. Cagaš, Extrakce lokálních obrazových deskriptorů na GPU, Brno, Masarykova univerzita, 2014 [Online]. Dostupné z: http://is.muni.cz/th/395960/fi_b/bakalarska_prace.pdf

[25] Významné body v obraze, Petr Bílek, České vysoké učení technické v Praze, Fakulta elektrotechnická, 2007 [Online]. Dostupné z: https://dip.felk.cvut.cz/browse/pdfcache/bilekp3_2007bach.pdf

[26] J. Šváb, Akcelerace zpracování obrazu hradlovým polem, Praha, 2009 [Online]. Dostupné z: http://cyberold.felk.cvut.cz/research/theses/papers/60.pdf

[27] Computer vision talks, 2011 [Online]. Dostupné z: http://computer-vision-talks.com/2011-08-19-feature-descriptor-comparison-report/

15 Příloha A – Manuál k aplikacím

15.1 Vytvoření databáze

Pro ukládání dat lze zvolit MySQL databázový systém dostupný z www.mysql.cz, nebo PostgreSQL databázový systém dostupný www.postgresql.org

Pro vytvoření MySQL databáze vložte následující skript do konzole a spusťte.

```
CREATE DATABASE `image_detection_db` USE `image_detection_db`;


CREATE TABLE `config` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(32) COLLATE utf8_czech_ci NOT NULL,
  `value` varchar(64) COLLATE utf8_czech_ci NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

INSERT INTO `config` (`id`, `name`, `value`) VALUES
(1,       'IMAGE_PATH',       '');


CREATE TABLE `image` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `name` varchar(150) COLLATE utf8_czech_ci NOT NULL,
  `description` text COLLATE utf8_czech_ci NOT NULL,
  `file_name` varchar(150) COLLATE utf8_czech_ci NOT NULL,
  `date_created` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

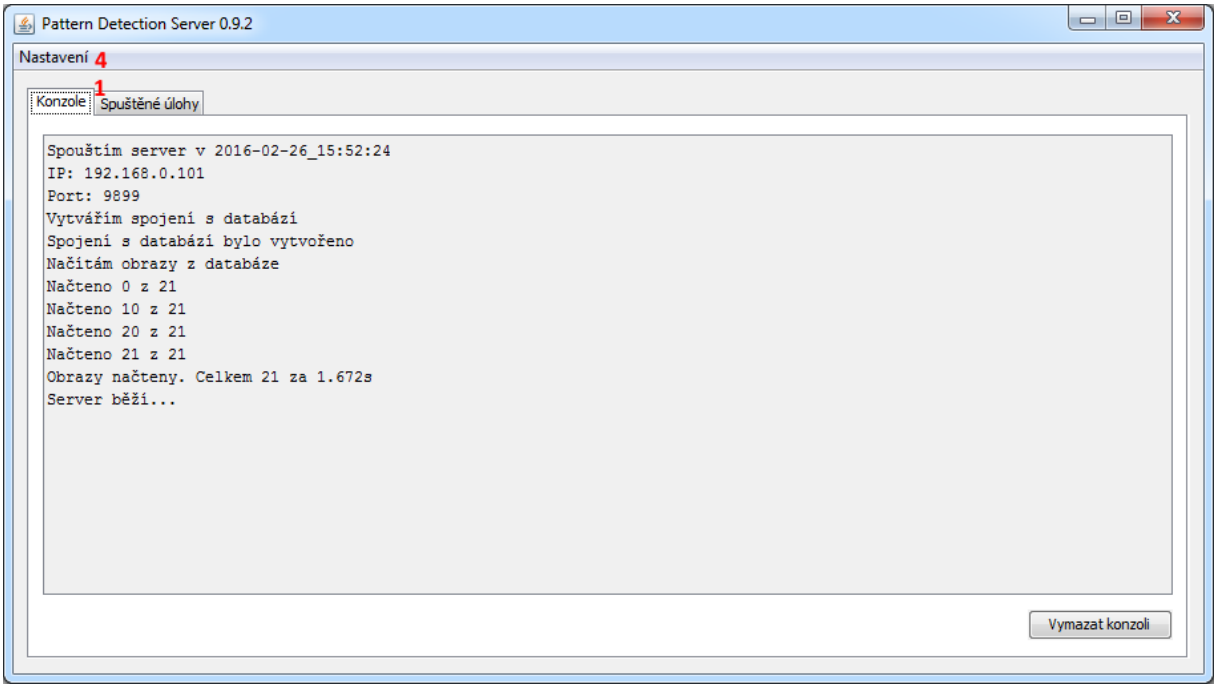
Pro vytvoření PostgreSQL databáze vložte následující skript do konzole a spusťte.

```
CREATE DATABASE image_detection_db


CREATE TABLE config
(
  id serial NOT NULL,
  name text,
  value name,
  CONSTRAINT config_pkey PRIMARY KEY (id)
)

INSERT INTO config (name) VALUES ('IMAGE_PATH');


CREATE TABLE image
(
  name text,
  description text,
  id serial NOT NULL,
  file_name text,
  date_created timestamp with time zone,
  CONSTRAINT image_pkey PRIMARY KEY (id)
)
```

15.2 Serverová aplikace

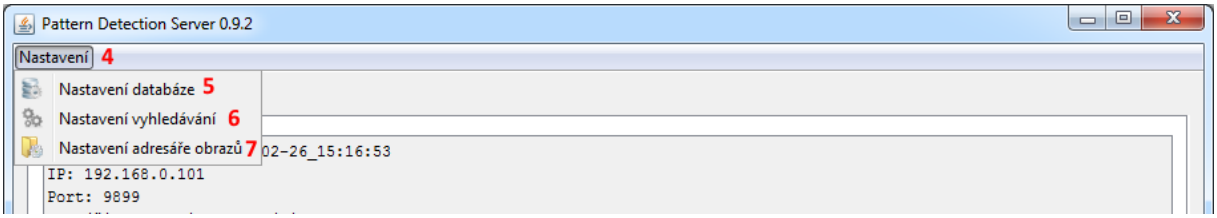
Serverová aplikace slouží pro vyhledávání obrazů z databáze. Přestože aplikace běží na serveru a běžně se k ní uživatel nedostane, obsahuje jednoduché uživatelské rozhraní. To slouží především k nastavení aplikace a zobrazování událostí, které na serveru proběhly. Grafické uživatelské rozhraní vypadá jako na obrázku A1, A2 a A3.

Skládá se z dvojice záložek (1) „Konzole“ a „Spuštěné úlohy“. Pod záložkou konzole se zobrazují události, které na serveru proběhly. Těmi může být připojení klienta, spuštění a ukončení vyhledávání, přidávání nového obrazu do databáze apod. Druhá záložka „Spuštěné úlohy“ zobrazuje právě probíhající hledání (2). Tyto úlohy lze v případě nouze ukončit tlačítkem „Ukončit“ (3).

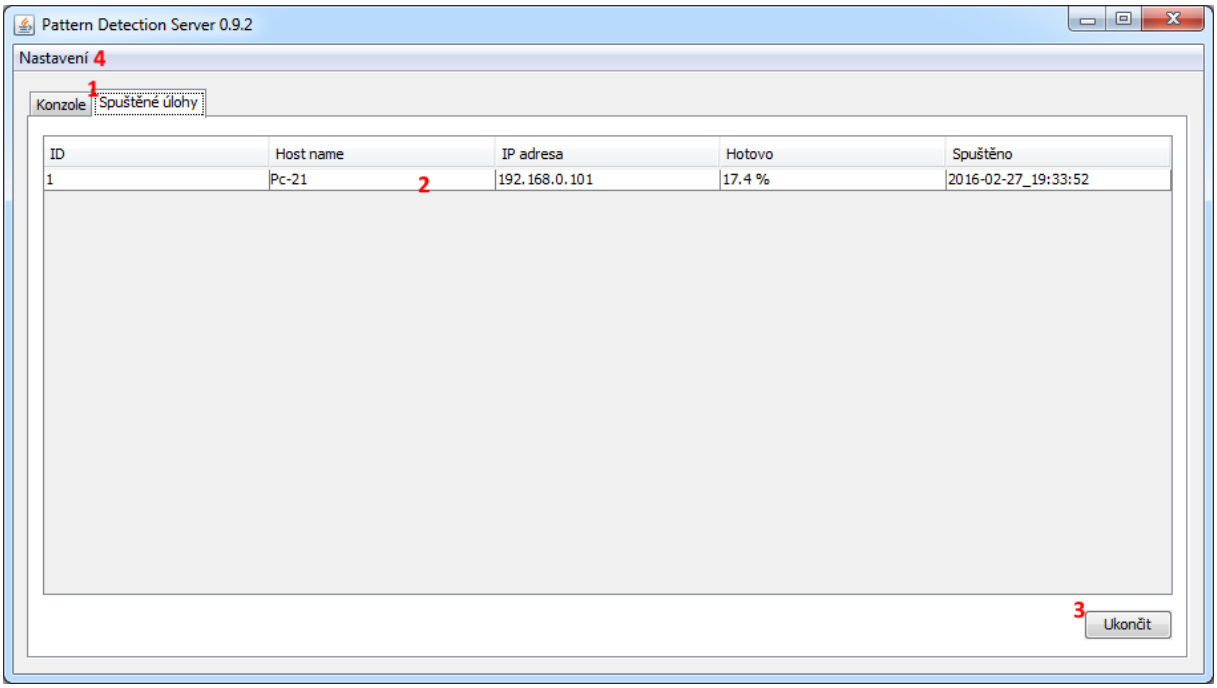


Obrázek A1 Grafické uživatelské rozhraní serverové aplikace – Konzole

Aplikace dále obsahuje hlavní menu s jedním tlačítkem „Nastavení“ (4). To obsahuje tři položky. „Nastavení databáze“ (5), „Nastavení vyhledávání“ (6) a „Nastavení adresáře obrazů“ (7), viz obrázek A2.

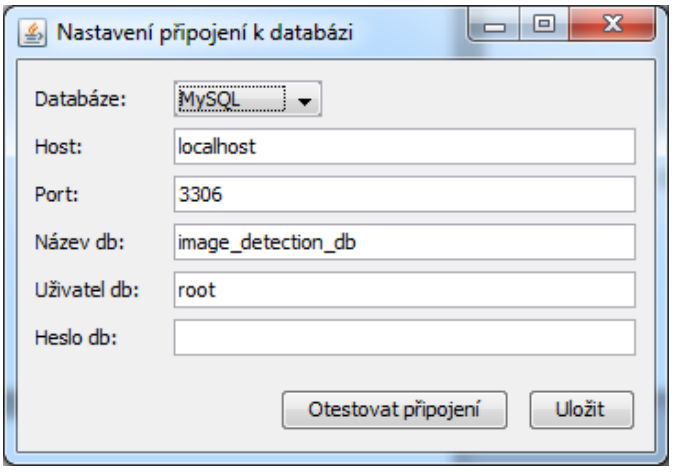


Obrázek A2 Grafické uživatelské rozhraní serverové aplikace – Nastavení



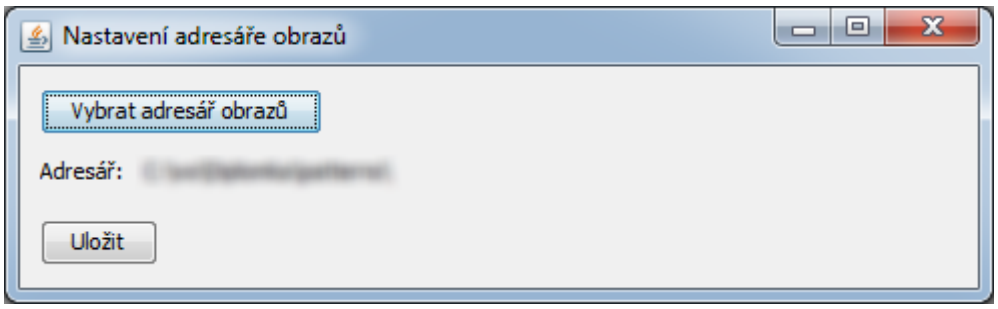
Obrázek A3 Grafické uživatelské rozhraní serverové aplikace – Spuštěné úlohy

Aby aplikace správně fungovala, je nutné ji nejprve správně nastavit. Prvním krokem je nastavení připojení k databázi pomocí položky „Nastavení databáze“ (5). Po vybrání této položky se zobrazí okno, jako je na obrázku A4. Nastavte typ databáze, kterou používáte. Do kolonky Host vyplňte IP adresu počítače, na kterém databáze běží. Pokud běží na stejném počítači jako tato aplikace, můžete ponechat „localhost“. Dále vyplňte kolonku Port podle typu databáze. Standardně pro MySQL je port 3306 a pro PostgreSQL je port 5432. Dále vyplňte název databáze pro ukládání obrazů a přihlašovací jméno a heslo pro připojení k databázi.



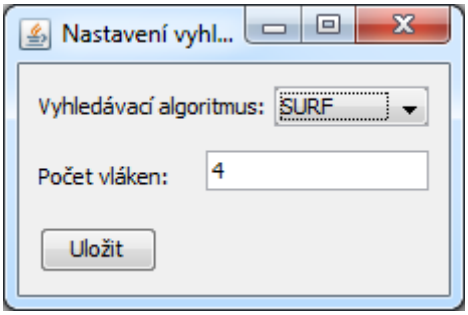
Obrázek A4 Nastavení připojení k databázi

Druhým krokem je nastavení adresáře pro ukládání obrazů pomocí položky „Nastavení adresáře obrazů“ (7). Po vybrání této položky se zobrazí okno, jako je na obrázku A5. Do vybraného adresáře se poté stahují všechny obrazy, které se přes administraci přidávají.



Obrázek A5 Nastavení připojení k databázi

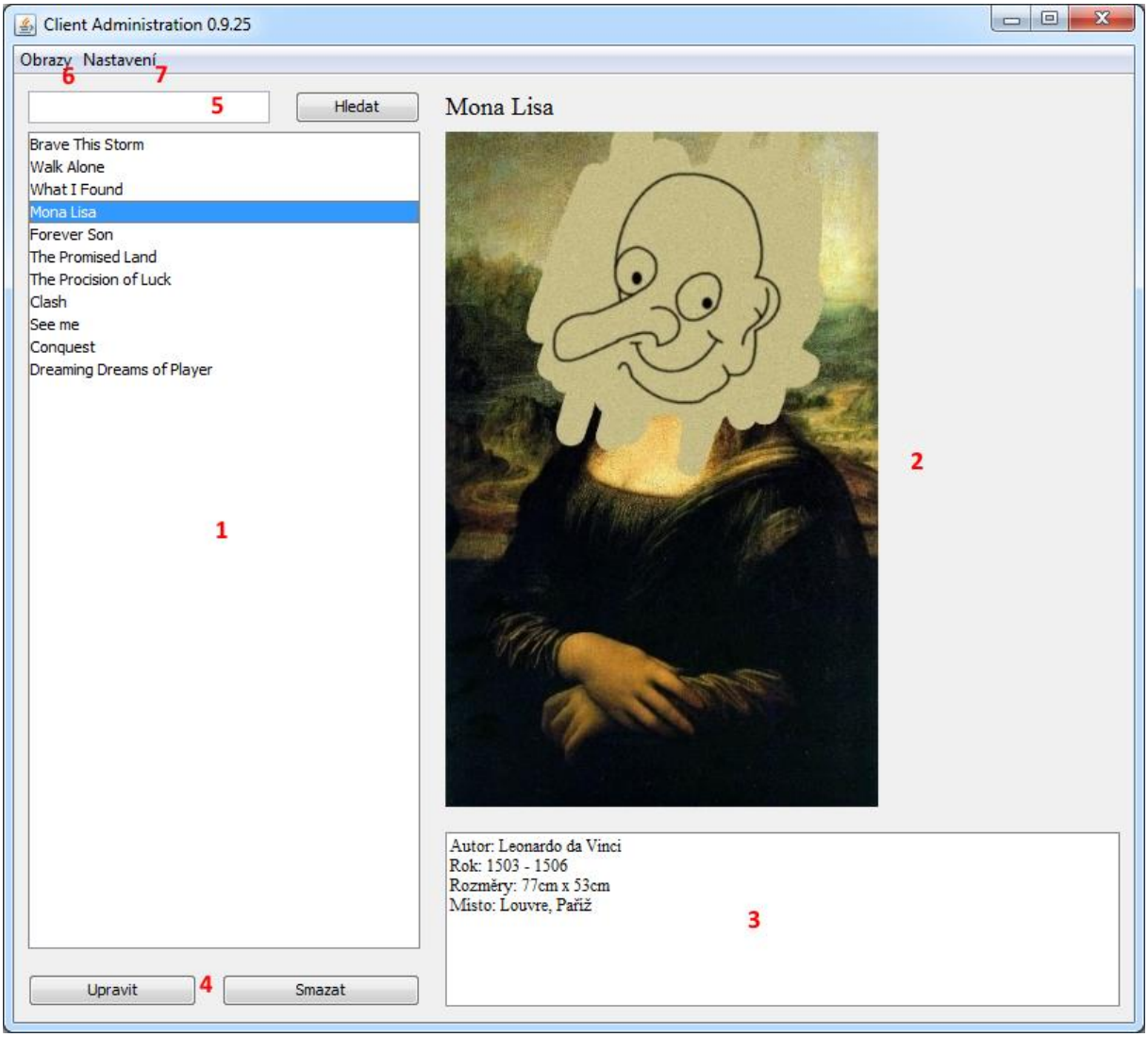
Posledním krokem je nastavení vyhledávacího algoritmu. To lze provést pomocí položky „Nastavení vyhledávání“. Po vybrání této položky se zobrazí okno, jako je na obrázku A6. Vybrat lze mezi algoritmy SURF (Speeded up robust features) a GHT (generalised Hough transform). Dále je také možné vybrat, na kolika vláknech bude vyhledávání probíhat. Od tohoto okamžiku je aplikace připravena pro vyhledávání.



Obrázek A6 Nastavení vyhledávání

15.3 Administrátorská aplikace

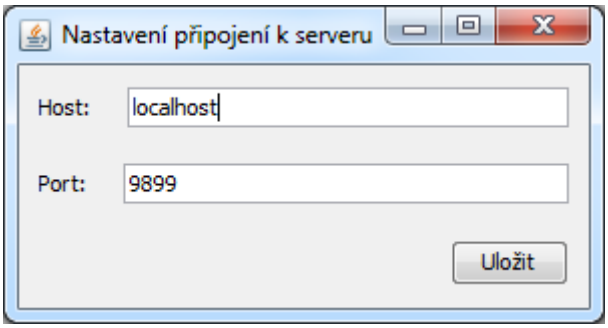
Tato aplikace má na starosti správu obrazů, mezi kterými se vyhledává. Grafické uživatelské rozhraní je možné vidět na obrázku A7. Skládá se ze seznamu obrazů (1), volné plochy, ve které se zobrazují vybrané obrazy (2) ze seznamu a pod tímto místem se nachází prostor pro výpis metadat o vybraném obrazu (3). Obrazy lze upravovat nebo mazat pomocí tlačítek v levé dolní části aplikace (4). Mezi obrazy lze také vyhledávat dle názvu (5), pokud je vyhledávací pole prázdné, jsou načteny všechny obrazy. V horní části aplikace se nachází hlavní menu, ve kterém jsou dvě tlačítka. Jedno pro přidávání obrazů (6) a druhé pro nastavení (7).



Obrázek A7 Grafické uživatelské rozhraní administrátorské aplikace

Prvním krokem při spuštění aplikace je nastavení připojení k serveru. To se provádí přes tlačítko „Nastavení“ v horní části aplikace (7), kde se nachází pouze jedna položka

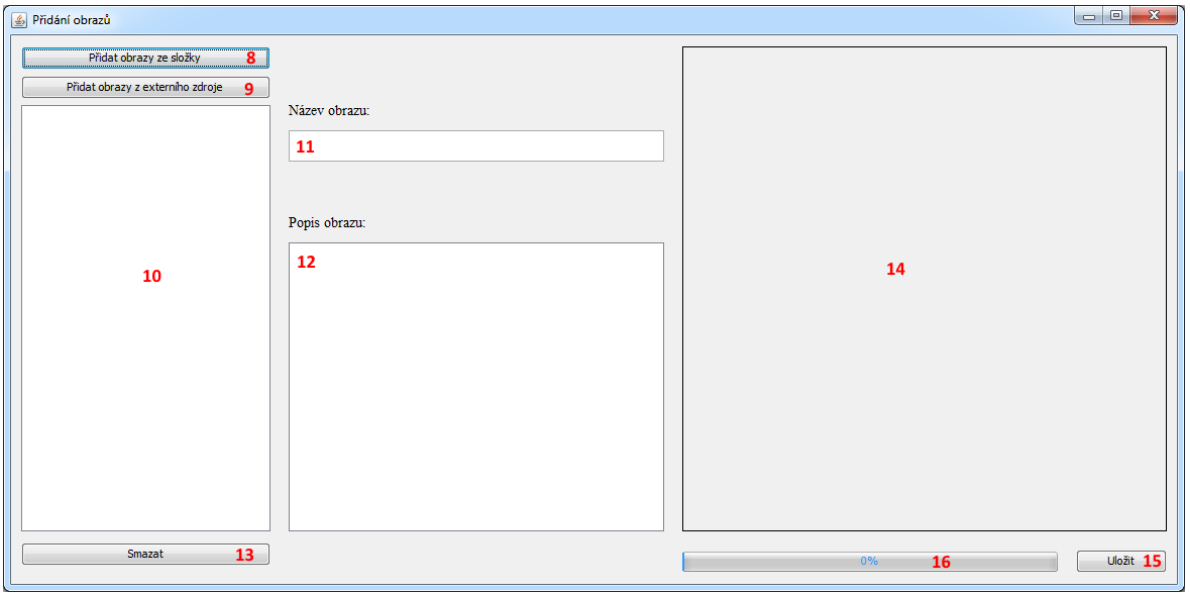
„Nastavení připojení k serveru“. Po vybrání této položky se zobrazí okno, jako je na obrázku A8. Do kolonky Host se vyplní IP adresa serveru a do kolonky Port číslo portu, na kterém serverová aplikace běží.



Obrázek A8 Nastavení připojení k serveru

Pokud je vše vyplněno správně, aplikace okamžitě naváže spojení se serverem a načte seznam obrazů z databáze. V opačném případě se zobrazí příslušná chybová hláška, a to že se spojení nepodařilo navázat.

Přidávat obrazy do databáze lze pomocí tlačítka v horní části menu „Obrazy“ (6). Zobrazí se okno jako na obrázku A9. Obrazy lze vybírat z lokálního umístění na disku pomocí tlačítka „Přidat obrazy ze složky“ (8) nebo také z externího zdroje pomocí URL. Ta se vkládá do okna, které se zobrazí po kliknutí na tlačítko „Přidat obrazy z externího zdroje“ (9). Všechny obrazy, které se vyberou pro přidání, se zobrazují v seznamu v levé části aplikace (10). Obrazy lze procházet a měnit jejich metadata, jako je název (11) a jejich popis (12), případně lze obrazy odebírat pomocí tlačítka „Odebrat“ (13). Vybraný obraz ze seznamu se zobrazuje v pravé části aplikace (14). Po skončení přidávání obrazů tlačítkem „Uložit“ (15) se začnou obrazy ukládat. Postup odesílaní je znázorněn progress barem vedle tlačítka „Uložit“ (16). Po skončení ukládání se nové obrazy zobrazí v seznamu obrazů (1).



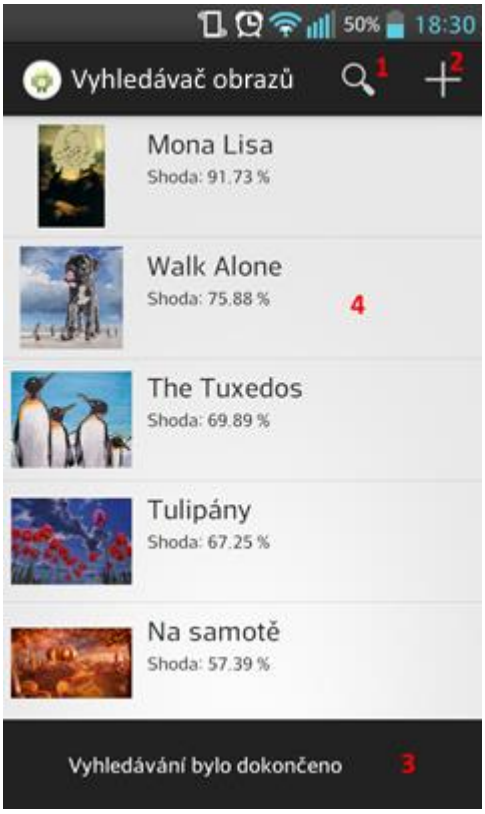
Obrázek A9 Grafické uživatelské rozhraní pro přidávání obrazů

Pro úpravu obrazu slouží podobné okno jako pro přidávání obrazů. Do něho se lze dostat pomocí tlačítka „Upravit“ (4), nebo dvojitým poklepáním na název obrazu ze seznamu (1). Zobrazí se okno jako na obrázku A10. Pomocí dvojice tlačítek v levé horní části obrazovky lze vložit nový obraz buď ze složky z lokálního umístění (17), nebo z externího zdroje pomocí URL adresy (18). Vybraný obraz se poté zobrazí v pravé části aplikace (19). Také lze změnit metadata o obrazu jako je název (20) a popis obrazu (21).

Obrázek A10 Grafické uživatelské rozhraní pro úpravu obrazů

15.4 Klientská aplikace Android

Tato aplikace slouží pro hledání obrazů, na mobilních zařízeních s operačním systémem Android. Hlavní obrazovka vypadá jako na obrázku A11. Skládá se z tlačítka pro spuštění vyhledávání (1), tlačítka pro výběr vzoru pro hledání (2), notifikační oblasti (3) v dolní části obrazovky a zobrazovací plochy (4) pro zobrazení nalezených výsledků.



Obrázek A11. Hlavní obrazovka

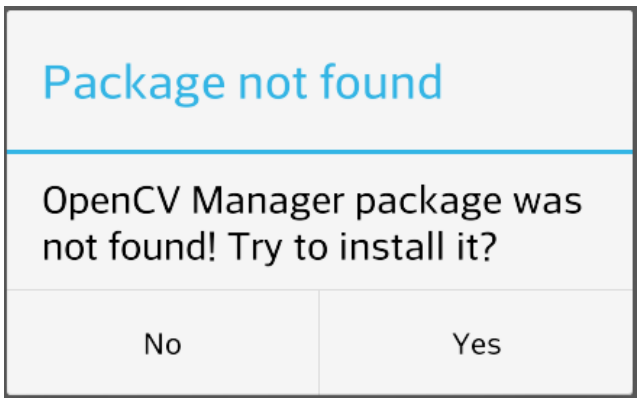
Obrázek A12. Hlavní obrazovka

Při prvním spuštění je třeba nastavit připojení k serveru. Po stisknutí tlačítka „Menu“ se zobrazí v dolní části obrazovky tlačítko „Nastavení“ (7) jako na obrázku A13. Po výběru tohoto tlačítka se zobrazí obrazovka pro nastavení připojení k serveru, jako je na obrázku A14. Zde je třeba zadat IP adresu serveru (9) a port (10), na kterém server běží. Správnost nastavení lze ověřit tlačítkem v pravé horní části obrazovky (8). Po jeho stisknutí se aplikace pokusí navázat spojení se serverem a během okamžiku se zobrazí informační zpráva, zda-li bylo spojení navázáno či nikoliv.

Aplikace vyžaduje pro fungování OpenCV Manager. Při prvním spuštění dochází k ověření, zda-li je tato aplikace nainstalovaná a pokud není, zobrazí se okno s výzvou pro instalaci, jako je na obrázku A15.

Obrázek A13. Tlačítko s nastavením

Obrázek A14. Obrazovka s nastavením

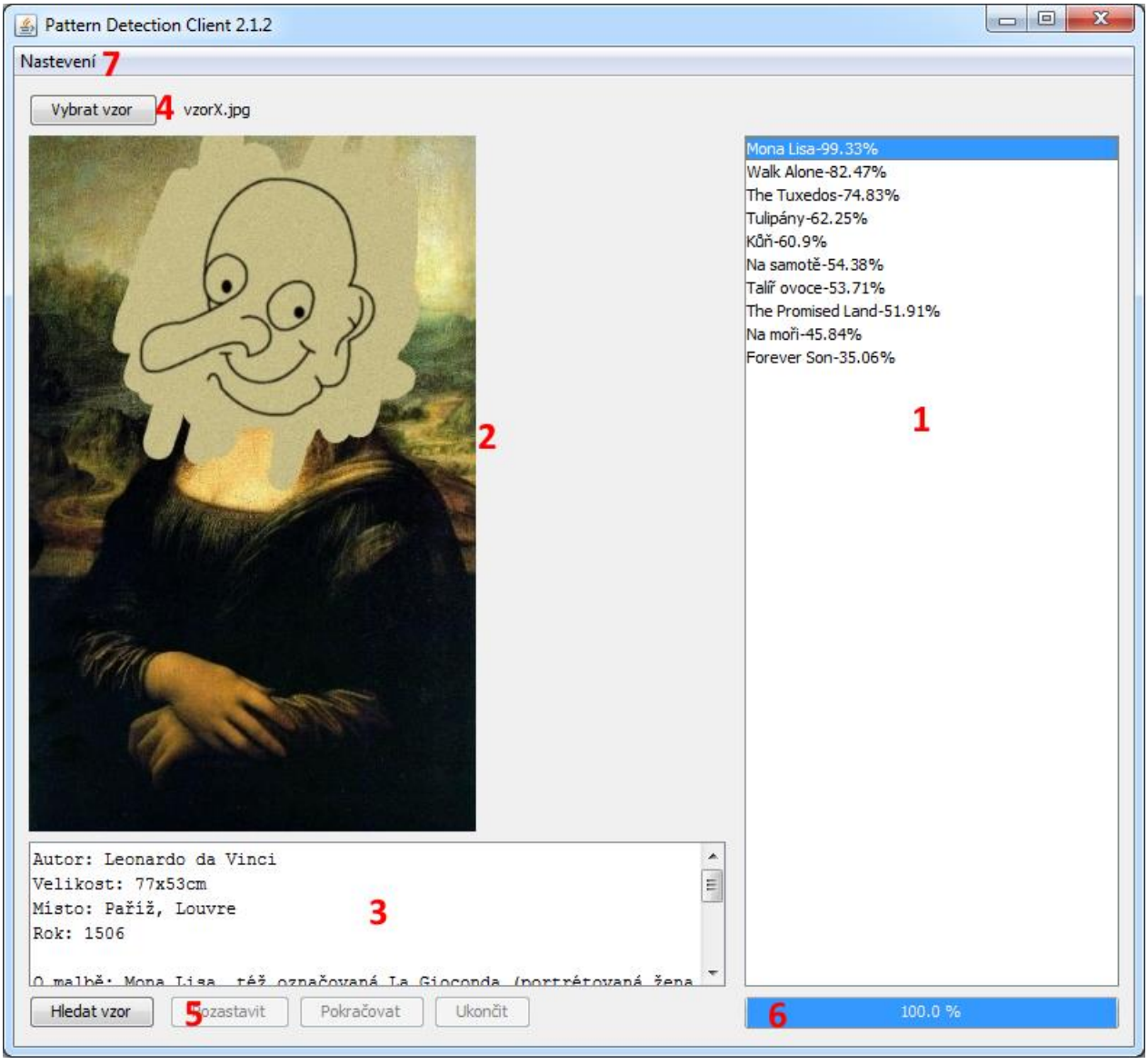


Obrázek A15. Výzva k instalaci OpenCV Manageru

Pro vyhledávání je třeba vybrat vzor. To lze provést pomocí tlačítka v pravé části obrazovky (2). Vzor lze načít z galerie mobilního zařízení (5), nebo ho lze pořídit pomocí fotoaparátu (6). Po jeho výběru se zobrazí v dolní části obrazovky notifikace, že obraz byl vybrán. Poté je možné spustit vyhledávání (1). Postupně se začnou zobrazovat výsledky. Pokud je vrácen požadovaný obraz ještě před skončením vyhledávání, je možné toto vyhledávání ukončit pomocí tlačítka, které se objevilo po spuštění vyhledávání v dolní části obrazovky. Pro každý obraz je možné zobrazit jeho detail.

15.5 Klientská aplikace pro počítače

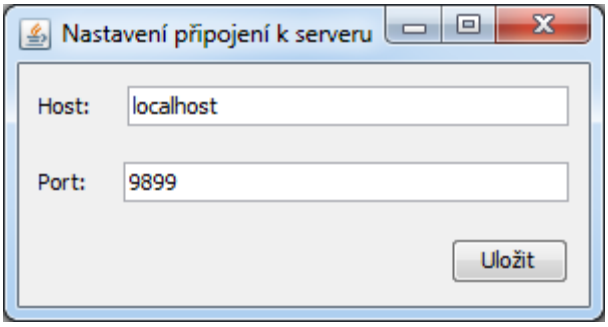
Tato aplikace slouží pro vyhledávání vzoru v databázi obrazů. Grafické uživatelské rozhraní vypadá jako na obrázku A16. Skládá se ze seznamu (1) v pravé části obrazovky, ve kterém se zobrazují nalezené výsledky. V levé části obrazovky (2) se zobrazuje obraz, který byl vybrán ze seznamu a pod ním se zobrazují příslušné informace (3). Vzor pro vyhledávání se vybírá pomocí tlačítka „Vybrat vzor“ (4). Samotné vyhledávání lze poté ovládat pomocí tlačítek (5) v dolní části obrazovky a jeho průběh je znázorněn progress barem (6).



Obrázek A16. Hlavní obrazovka

Při prvním spuštění je třeba nastavit připojení k serveru. To se provádí pomocí tlačítka v hlavním menu (7) v horní části obrazovky. Po vybrání se zobrazí okno A17. Do kolonky

Host se vyplní IP adresa serveru a do kolonky Port číslo portu, na kterém serverová aplikace běží. Poté je aplikace připravena.



Obrázek A17. Obrazovka pro nastavení

Pro vyhledávání poté vyberte vzor pomocí tlačítka „Vybrat vzor“ (4). Vybraný vzor se poté zobrazí v prázdné oblasti (2) pod tlačítkem. Hledání se spustí tlačítkem „Hledat vzor“. Průběžně se začnou zobrazovat mezivýsledky v seznamu (1). Pokud se Vámi hledaný vzor zobrazí ještě před dokončením vyhledávání, je ho možné pozastavit, nebo úplně ukončit pomocí příslušných tlačítek v dolní části aplikace (5).

16 Příloha B – Obsah přiloženého CD

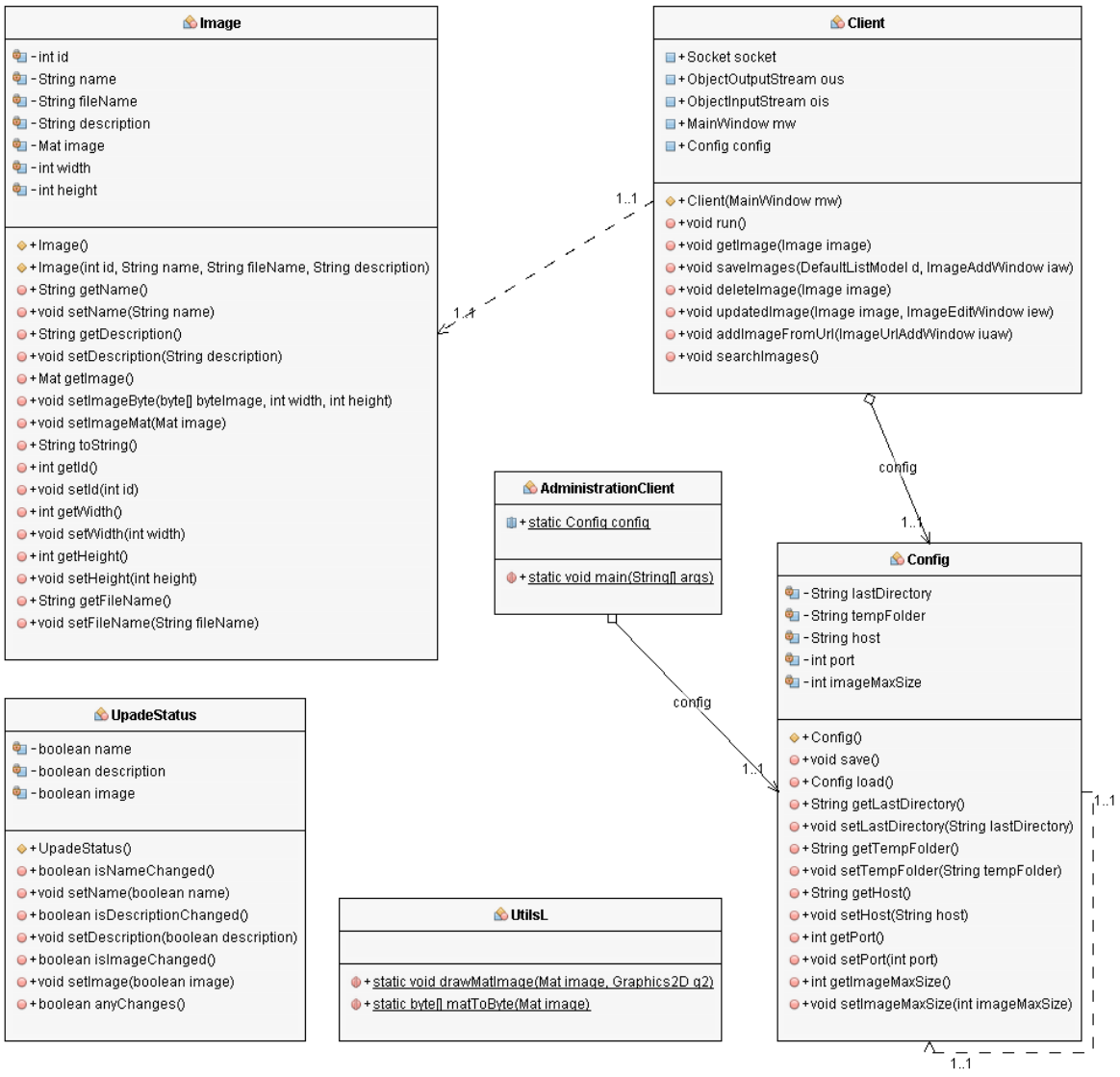
- text diplomové práce ve formátu pdf
- obrazy použité pro testování ve formátu jpg
- vzory použité pro testování ve formátu jpg

- zdrojové kódy serverové aplikace
- spustitelný soubor serverové aplikace
- zdrojové kódy administrátorské aplikace
- spustitelný soubor administrátorské aplikace
- zdrojové kódy klientské aplikace pro Android
- spustitelný soubor klientské aplikace pro Android
- zdrojové kódy klientské aplikace pro počítače
- spustitelný soubor klientské aplikace pro počítače
- zdrojové kódy knihovny

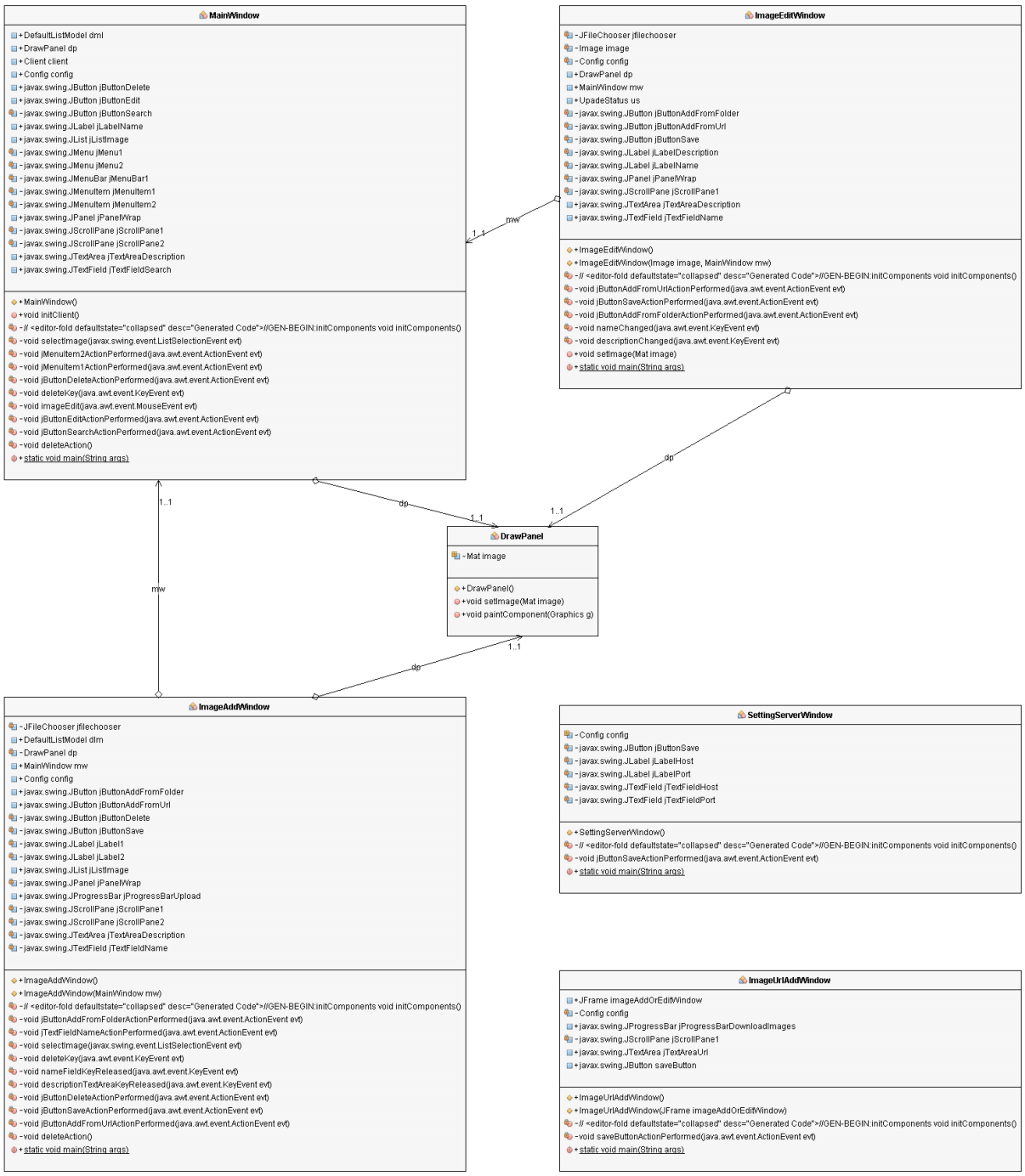
17 Příloha C – Class diagramy

V této příloze se nachází class diagramy všech vytvořených aplikací. Obrázky ve větším rozlišení jsou k dispozici na přiloženém CD ve složce „Class diagramy“.

17.1 Class diagramy administrátorské aplikace



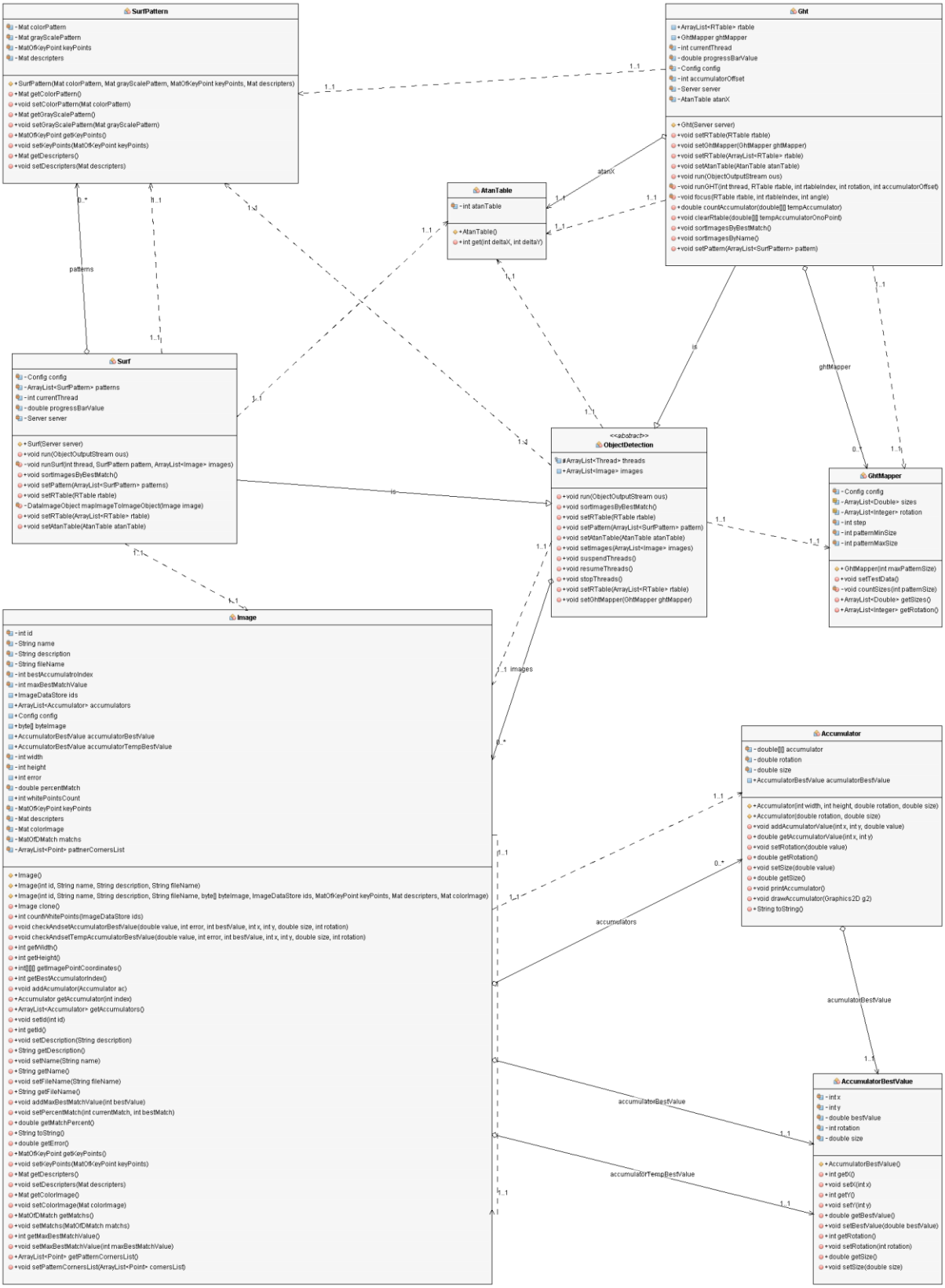
Obrázek C.1 Class diagram administrátorské aplikace – administration.core



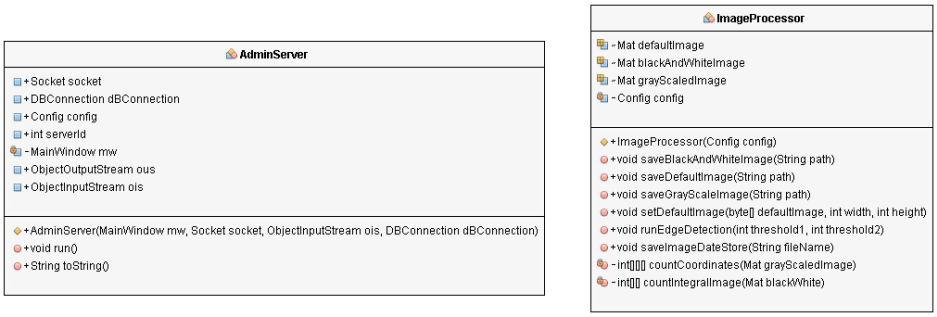
Obrázek C.2 Class diagram administrátorské aplikace – administration.gui

17.2 Class diagramy serverové aplikace

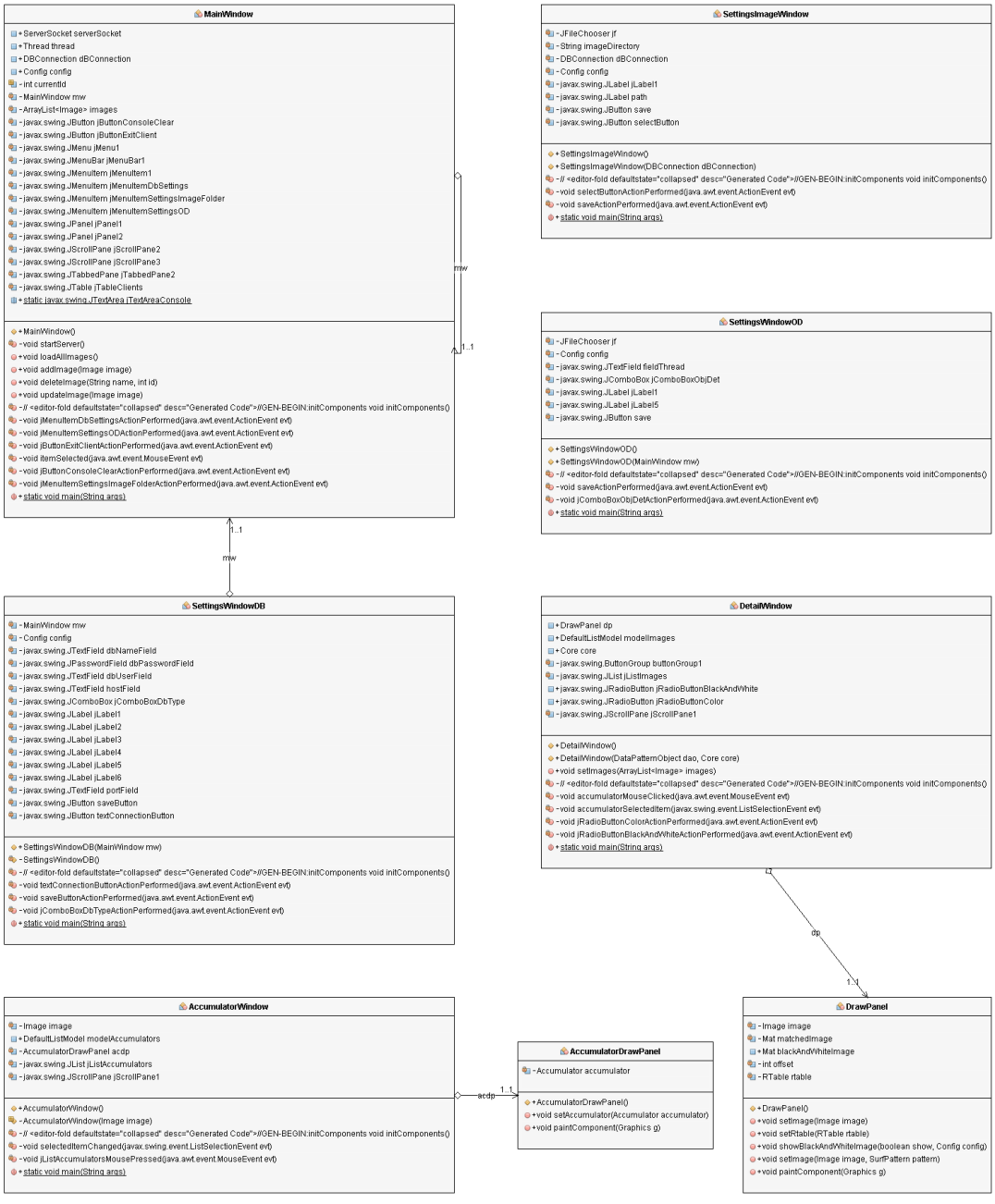
Obrázek C.3 Class diagram serverové aplikace – balíček server.core



Obrázek C.4 Class diagram serverové aplikace – balíček server.objectdetection

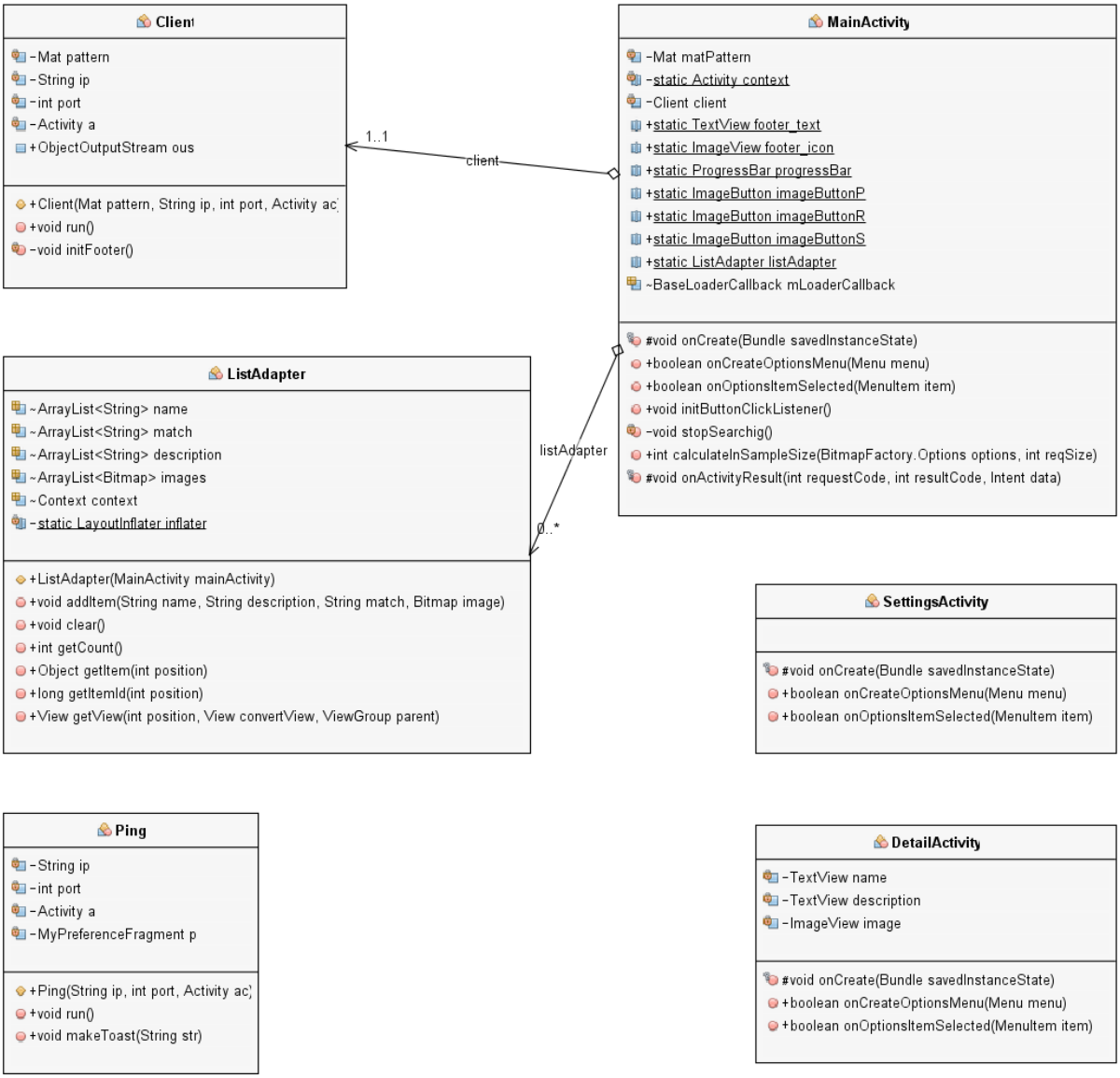


Obrázek C.5 Class diagram serverové aplikace – balíček server.admin



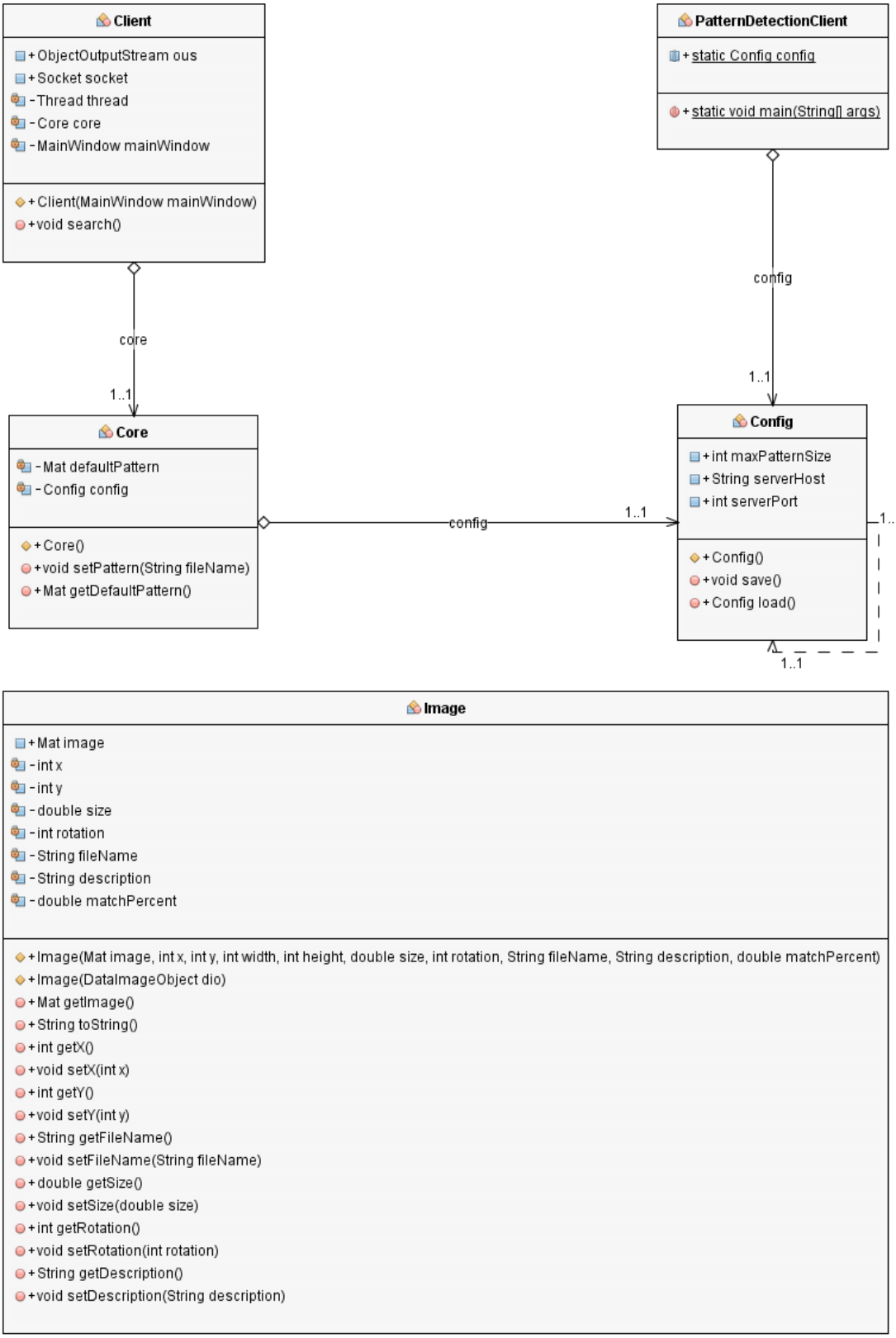
Obrázek C.6 Class diagram serverové aplikace – balíček server.gui

17.3 Class diagram klientské aplikace pro Android

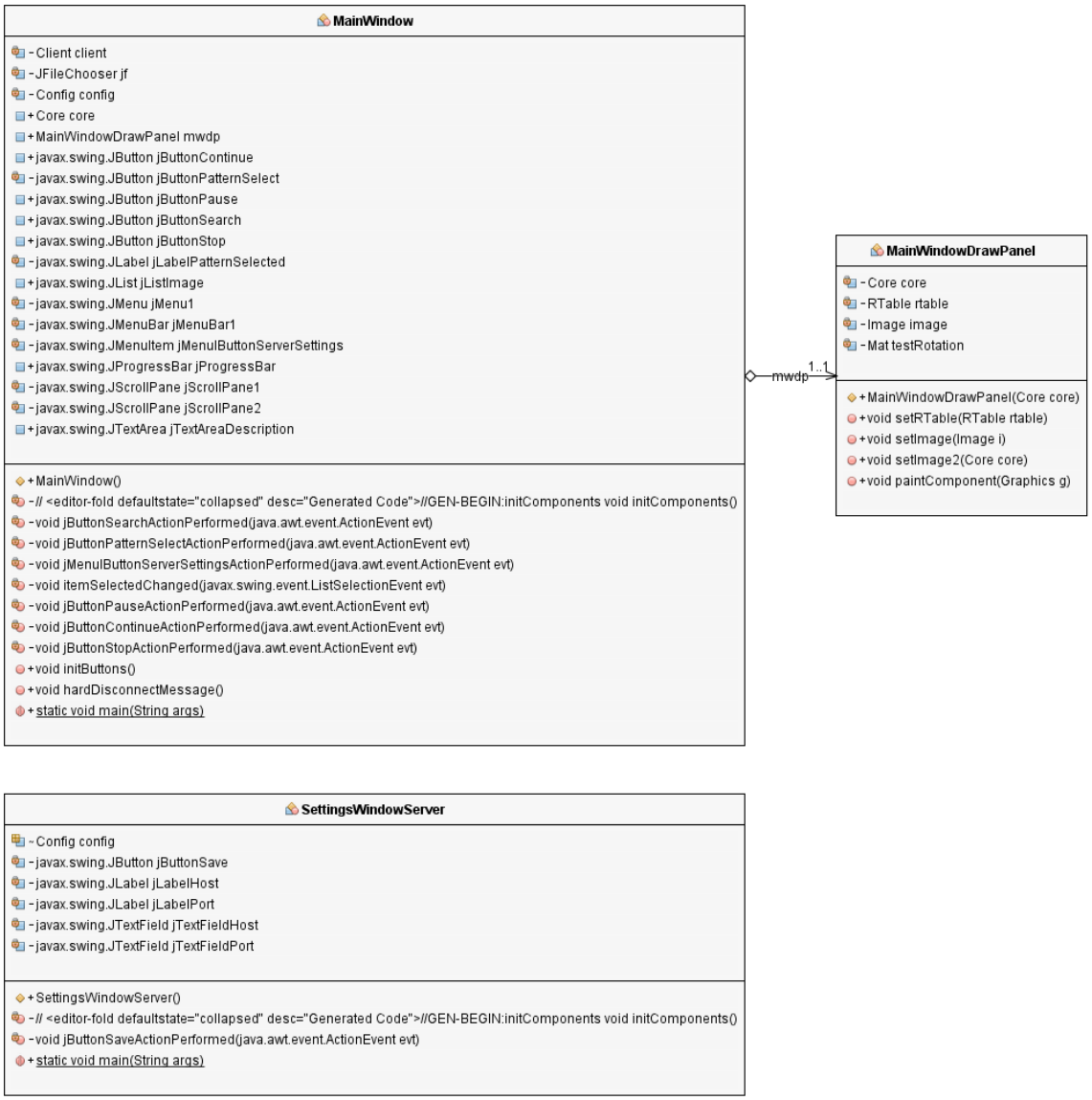


Obrázek C.7 Class diagram klientské aplikace pro Android

17.4 Class diagramy klientské aplikace pro počítače

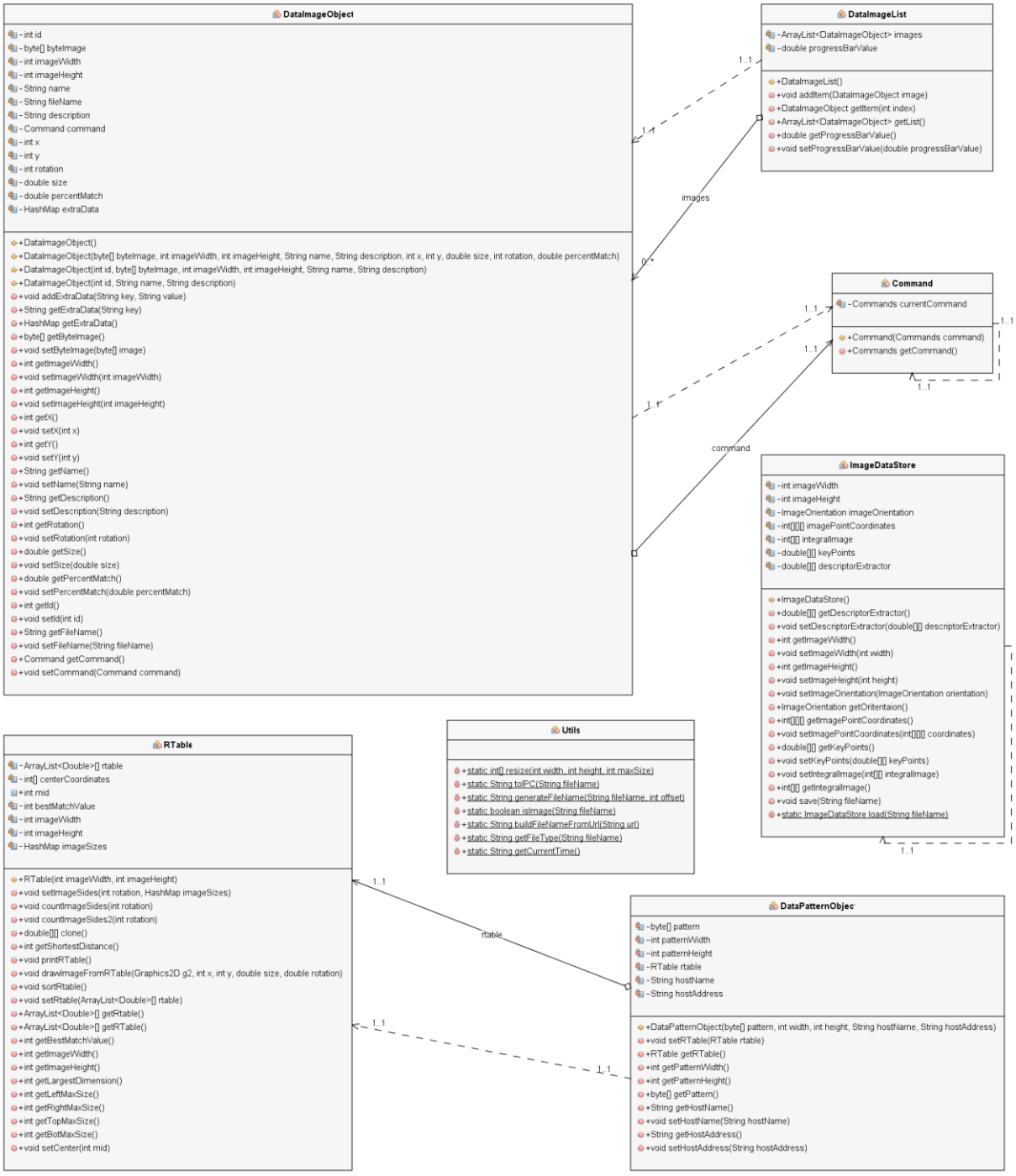


Obrázek C.8 Class diagram klientské aplikace pro počítače – balíček client.core



Obrázek C.9 Class diagram klientské aplikace pro počítače – balíček client.gui

17.5 Class diagram knihovny



Obrázek C.10 Class diagram knihovny